A: INTRODUCTION

Object Recognition and Image Classification has been an old and active area of research in Machine Learning community, with new theories and research published every now and often, pushing the state-of-the-art limits. Beginning with text-based image classification and followed by visual features based object recognition, we have reached a stand where we can directly recognize the objects in an image from the pixel values of the image itself, with a note that this feat being done in seconds on powerful Graphics Processing Unit (GPU) based system. The goal of computer vision enthusiasts has always been to emulate human vision system, and technological advancements in hardware, as well as software, have shown that we are near to perfection.

Image processing also forms an inevitable part of Computer Vision which deals with enhancing images, extracting informative and discriminative visual features, dimensionality reduction, etc. which facilitates the image classification process. Visual features are mainly handcrafted, i.e. they are mathematical formulae which capture pixel properties of images, for example, color histograms, wavelet textures, edge direction histogram, etc. Few past decades has seen a significant technological advancement in visual features representation, which is interestingly not handcrafted but is automatically learned from images such that those visual features are best suited for the task they are to be used for. Such visual features are learned from Deep Learning (DL) based technologies, mainly Convolutional Neural Networks (CNNs). Handcrafted features are commonly called as Shallow features and visual features extracted with DL libraries are called as Deep features. Recent research has shown that Deep features perform better than shallow features.

Coming to the image classification literature, the classical setting is to classify similar images as present in training database. For example, if we have a database of images belonging to classes Elephant, Tiger, Human, Chair, Dog, Car, and Aeroplane, then the database is partitioned into training and testing dataset, with images belonging to a joint set of classes mentioned above. So the classification algorithm learns to identify images from training dataset and then classifies similar images during the testing phase. However a recent image classification setting has been devised (not even a decade old at the time of writing this thesis), called Zero-Shot Learning framework, where the images in training and test set belong to disjoint classes. For example, training set would correspond to images belonging to classes Elephant, Human, Dog, Chair and test set would correspond to images belonging to classes Tiger, Car and Aeroplane. On the first instance, this problem looks intractable, and it actually is, if we address this problem with state-of-the-art classical approaches of image classification. So we need an informative and discriminative subspace through which information transfer can take place. Another interesting variation of image classification is an amalgam of Zero-Shot Learning and Classical Image Classification, called One-Shot Learning, where one or a very few limited number of training images of test set classes are present during training time.

This Master's Thesis report presents algorithms for the above mentioned different settings of image classification through subspace learning. A subspace is defined as a space that is wholly contained in another space, or whose points or elements are all in another vector space. Subspace learning is used for the better representation of multiple visual features as seen in section (Informed Subspace Learning) or for discriminative information transfer as seen in section (Zero-Shot Learning). Two approaches of subspace learning have been described below:

a) Matrix Factorization Based Subspace Learning (Data Fusion)

This method typically deals with classical image classification setting where training and test set classes are joint. As the name suggests, it is a method to break down a matrix into two factors, namely a component matrix and a basis matrix. Here, this method is used to fuse the information present in multiple matrices, resulting in a matrix which is a latent subspace of combining matrices and contains better discriminative information. This fusion of information is also known as Data Fusion, which is broadly classified into two sub-categories: Early Fusion and Late Fusion. The matrix factorization method discussed here is used for Early Fusion, where we fuse the raw information i.e. multiple types of visual features of images.

b) Intermediate Semantic Space

This subspace or space is used for Zero-Shot Learning framework as explained earlier, a setting where the training and test classes are disjoint. We need a vector space through which discriminative information extracted from training dataset is transferred to test dataset. We shall see that the intermediate semantic space for Zero-Shot Learning is constructed from Word Vectors as well as with high-level n-dimensional Attributes which are human distinguishable features. Since the classes are disjoint, i.e. no or zero training samples of test classes are present during training time, so it is named as Zero-Shot Learning.

In further chapters, research work done to address both the classical and novel image classification problem, namely Learning an Informative Subspace for Image Classification from multiple available features in joint class setting and Learning an Intermediate Semantic Space for Zero-Shot Learning in disjoint class setting respectively are discussed.

B. INFORMATIVE SUBSPACE LEARNING VIA SUPERVISED MATRIX FACTORIZATION

ABSTRACT

Matrix factorization technique has been widely used as a popular method to learn a joint latent-compact subspace when multiple views or modals of objects (belonging to a single domain or multiple domains) are available. Our work confronts the problem of learning an informative latent subspace by imparting supervision to matrix factorization for fusing multiple modals of objects, where we devise simpler supervised additive updates instead of multiplicative updates, thus scalable to large-scale datasets. To increase the classification accuracy, we integrate the label information of images with the process of learning a semantically enhanced subspace. We perform extensive experiments on two publicly available standard image datasets of NUS-WIDE Object and NUS-WIDE Scene and compare the results with state-of-the-art subspace learning and fusion techniques to evaluate the efficacy of our framework. The improvement obtained in the classification accuracy confirms the effectiveness of our approach. In essence, we propose a novel method for supervised data fusion thus leading to supervised subspace learning.

Keywords Image Classification, Joint Subspace Learning, Supervised Matrix Factorization

1. INTRODUCTION

Extensive research in the field of image retrieval furnish us with diverse techniques and literature on efficient image classification and retrieval, Content-Based Image Retrieval (CBIR) being one of them. Given a query image, we can easily obtain similar images from an image database using various fast and effective techniques of CBIR [7]. Most of the CBIR techniques exploit visual content (such as color, texture, spatial layout, etc.) of an image to retrieve a set of semantically similar images. To include supervision to this task, recently image retrieval systems have incorporated user's relevance feedback as a means to improve retrieval accuracy [23]. Multiple low-level visual features like gist features, color histograms, wavelet texture features, etc. characterize an image. Subspace learning technique is one of the popular methods for CBIR which learns a latent semantic subspace from these available multiple visual features. Various techniques of subspace learning include Principal Component Analysis (PCA) [3], Partial Least Squares (PLS) [24], Bilinear Model (BLM) [25], Linear Discriminant Analysis (LDA) [3], Locality Preserving Projection (LPP) [14] and Neighborhood Preserving Embedding (NPE) [15]. In [33] authors learn the shared subspace through the combination of CCA and Multi-Output Regularized Feature Projection.

An appropriate subspace not only integrates semantic knowledge from the individual features but also produces a combined compact representation for them. As a result, subspace learning techniques are widely used as a part of dimensionality reduction techniques where the intention is to extract out maximum discriminative information from multiple higher dimensional vector spaces [14]. There are a few other research directions where the subspace learning is accompanied with other tasks too, such as subspace learning with feature selection [11].

A general technique for subspace learning is matrix factorization for which there are two basic frameworks. One framework enforces the non-negativity constraints on the factors, while the other does not. For first framework rich literature is available for the subspace learning with Nonnegative Matrix Factorization (NMF) [19], [13] and references therein. To achieve better classification accuracy with NMF various extensions of the basic matrix factorization are proposed, for example, utilizing prior knowledge in the form of the inherent pattern in the dataset via structured regularizers [12], [31], [34], convex relaxation, etc. When more than one feature-sets of objects are available, the general mechanism of subspace learning is extended to the shared subspace learning, for which various NMF techniques are proposed. In [22], authors proposed an NMF-based multiview clustering algorithm that searches for a factorization that achieves better clusters across multiple views. A similar work is also proposed in [4]. (Terms "modals" and "feature-sets" are used interchangeably.)

As a purely unsupervised learning method, classic matrix factorization does not integrate any supervision and thus discards label information. Recently few authors proposed to utilize the supervision provided by class labels during matrix factorization to achieve better discriminative subspaces [30], [8], [21], [9], [17], [35]. In a very recent work [20] authors develop an NMF based model for cross-modality hashing and impart supervision to it by utilizing label information. These works above use multiplicative updates in their iterative process, which adds run-time complexity when performed on large scale datasets. This motivates us to design a new framework which not only utilizes the label information but is also applicable to large-scale datasets. Authors in [5] suggested using the gradient descent technique to model additive updates for the unsupervised matrix factorization method to solve the subspace learning problem.

In this paper, we propose a supervised matrix factorization technique (without non-negativity constraints on factors) to learn an informed latent subspace from multiple available visual feature-sets using iterative additive updates. Specifically, instead of multiplicative updates as is the case with NMF, we learn supervised additive update rules which scales well for large-scale datasets. For learning an informed subspace with more discriminating ability and improved classification performance, we exploit the label information of images synchronously with matrix factorization where it is embedded iteratively using additive updates in the subspace learning process. In an independent and very recent work [32] authors learn a single view classifier for multi-view data where their optimization method bears similarity with ours. They compose their objective function as a combination of Cauchy error estimator and hinge loss for learning intact feature vectors corresponding to each data point and designing a binary classifier in intact space which minimizes the classification error respectively.

However, our proposed framework has significant differences compared to theirs. The formulation of our objective loss function is comparatively simpler, and we deal with multiclass data instead of binary. They include the label information of binary classes by minimizing the classifier loss measured by hinge loss function whereas we integrate the label information by modeling a Linear Programming problem. Also, the intelligence of our method lies in learning a new representation of the objects in which similar images (images of one class) are mapped closer to each other than the dissimilar images (images belonging to different classes).

Apart from viewing our work as addressing the problem of informed subspace learning, its other facet lies in approaches to data/information fusion too. Two broad categories exist for data fusion, namely Early Fusion and Late Fusion. Early Fusion consists of feature level fusion techniques like feature concatenation, state-of-the-art subspace learning methods; for example Canonical Correlation Analysis (CCA), Kernel Canonical Correlation Analysis (KCCA) [1], Generalized Canonical Correlation Analysis (GCCA) [16] [29], Regularized Generalized Canonical Correlation Analysis (RGCCA) [26] [27], BLM, PLS etc. Most of these available techniques are qualified to the exploitation of only two feature-sets (reason can be attributed to the fact that they were developed for fusing cross-modal datasets; for example image and text datasets). However, GCCA and RGCCA are modeled for fusing more than two feature-sets. Late Fusion encompasses state-of-the-art methods like decision level fusion, score level fusion and rank based fusion [2].

To evaluate our proposed model, we apply our algorithm on two real image datasets and compare it with mentioned state-of-the-art subspace learning and fusion techniques along with the unsupervised counterpart of it. The resulting supervised subspace obtained by our proposed method shows better performance for image classification over the other compared methods. The outline of the paper is as follows. In the next immediate section, we discuss the problem with some preliminary concepts related to it. In Section 3, we formulate the mathematical model of our algorithm. The experimental results and analysis are presented in section 4. Finally, we conclude our work in Section 5.

2. PRELIMINARIES

Factorizing a matrix in two exact factors such that we retrieve the original matrix after multiplying them is an NP-hard problem [28]. To solve it, we tend to approximate the factors as shown in (1). Under the framework of unsupervised machine learning, matrix factorization process is used with different constraints to extract out the factors having different representational qualities. Most of the algorithms based on matrix factorization introduce the non-negativity constraints on factors. Consider a matrix $\mathbf{V} \in \mathbf{R}^{n \times m}$, the two factors obtained after NMF should be nonnegative, i.e.,

$$V \approx PH \quad (1)$$

Where **P** and **H** are non-negative ***n***×***d*** and ***d***×***m*** dimensional matrices respectively. The matrix **V** can be considered having ***m*** number of ***n***-dimensional vectors, where each column represents an entity. For instance, consider a set of ***m*** images where each image is represented by ***n***-dimensional feature-vector. Conceptually, the linear combinations of ***n*** rows of **P** approximate each entity vector ***v*** (column of **V**) weighted by ***h*** (corresponding column of **H**). The variable ***d*** is chosen smaller than both ***n*** and ***m*** to obtain factors with reduced dimension. Matrices **P** and **H** are referred as the basis matrix and the coefficient matrix respectively which can be found by formulating an optimization problem below,

$$\min_{P,H} \|V - PH\|_F^2,$$

where $\| \cdot \|_F^2$ is the Frobenius norm. The squared norm is minimized with respect to the factors **P** and **H** such that the approximated error in reconstructing the original matrix V is as minimum as possible. The squared norm term is the cost function which can be replaced by many available alternatives, for example, hinge loss or absolute loss. Consider the case where for a set of images more than one feature-sets are available, each contributing a specific low-level characteristic of the images. In this case, matrix factorization can be used to extract a latent unified shared subspace from these feature-sets that best represents the images by fusing the information contributed by each individual feature-set. Let $\mathbf{V}_1 \in \mathbf{R}^{n1\times m}$ and $\mathbf{V}_2 \in \mathbf{R}^{n2\times m}$ be two visual feature-sets, where n_1 and n_2 represent the dimension of visual features and ***m*** denotes the number of images. The joint matrix factorization can be given as,

$$V_1 \approx P_1 H$$
$$V_2 \approx P_2 H$$

The matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ are the two basis matrices corresponding to the two views and **H** is the coefficient matrix which we denote as the learned subspace shared by the feature-sets. The joint matrix factorization optimization problem for two feature-sets can be formulated as follows [5],

$$\begin{aligned} \min_{P_1,P_2,H} & \|V_1 - P_1 H\|_F^2 + \|V_2 - P_2 H\|_F^2 \\ & + \lambda(\|P_1\|_F^2 + \|P_2\|_F^2 + \|H\|_F^2), \end{aligned} \quad (2)$$

where λ is the regularization parameter that is used to control the loss minimization and the regularization function with respect to the three varying terms. In [5] iterative updates in terms of $\mathbf{P}_1$, $\mathbf{P}_2$ and **H** are given as follows,

$$\begin{aligned} P_1^{t+1} &\leftarrow P_1^t + \alpha((V_1 - P_1^t H^t){H^t}^T - \lambda P_1) \\ P_2^{t+1} &\leftarrow P_2^t + \alpha((V_2 - P_2^t H^t){H^t}^T - \lambda P_2) \\ H^{t+1} &\leftarrow H^t + \alpha({P_1^t}^T(V_1 - P_1^t H^t) + \\ &\qquad {P_2^t}^T(V_2 - P_2^t H^t) - \lambda H^t) \end{aligned} \tag{3}$$

It is easy to observe that the above formulation can be extended to any number of modals. However, the additive updates lack intelligence due to unsupervised matrix factorization which we tend to overcome by imparting supervision to the matrix factorization process. In

fact, through experiments, we show that our proposed approach outperforms this unsupervised method of subspace learning along with other state-of-the-art methods too. The next section discusses our model in detail.

3. PROPOSED FRAMEWORK

For a training set of $\boldsymbol{m}$ images, say $\mathbf{X} = \{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_m\}$,let $\mathbf{Y} = \{\boldsymbol{y}_1,\boldsymbol{y}_2,\cdots,\boldsymbol{y}_m\}$ be the set of labels for $\boldsymbol{m}$ images, where $\boldsymbol{y}_1,\boldsymbol{y}_2\cdots,\boldsymbol{y}_m \in\{1,\cdots,\mathbf{T}\}$ for $\mathbf{T}$ distinct classes. With the aim to improve the classification accuracy we integrate the label information with the process of subspace learning. We create two sets of pairs of training images, namely **S** and **D**. Set **S** comprises of all those pairs of images which belong to same class i.e. $\mathbf{S} = \{(\boldsymbol{x}_i,\boldsymbol{x}_j) : \boldsymbol{y}_i = \boldsymbol{y}_j,\ \boldsymbol{x}_i,\boldsymbol{x}_j \in \mathbf{X}\}$. Similarly, set **D** consists of pairs of all those images which belong to different classes, thus $\mathbf{D} = \{(\boldsymbol{x}_i,\boldsymbol{x}_j) : \boldsymbol{y}_i \mathrel{!=} \boldsymbol{y}_j,\ \boldsymbol{x}_i,\boldsymbol{x}_j \in \mathbf{X}\}$. This information of similarity and dissimilarity in between images in pairs, in set **S** and **D** respectively is utilized in the process of learning a distinctive and informative subspace. It is evident that a semantically rich representation of images should consist of image embeddings such that images belonging to a class are closer compared to images belonging to different classes. For achieving our mentioned latent subspace representation, the distance in between the images in pairs belonging to set **S** should be minimized, and the distance in between the images in pairs belonging to set **D** should be maximized. Considering the squared $\boldsymbol{l_2}$ norm as the distance metric, where $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ are vector representations of $\boldsymbol{i}$th and $\boldsymbol{j}$th image, we can write,

$$\min \sum_{i,j\in S} \alpha_{ij}\|h_i - h_j\|^2 \tag{4}$$

for images in pairs in set **S** and

$$\max \sum_{i,j \in D} \beta_{ij} \|h_i - h_j\|^2 \tag{5}$$

for images in pairs in set **D**. We can combine equations (4) and (5) with the minimization criteria as follows,

$$\min\{\sum_{i,j \in S} \alpha_{ij} \|h_i - h_j\|^2 - \sum_{i,j \in D} \beta_{ij} \|h_i - h_j\|^2\} \tag{6}$$

The parameters $\boldsymbol{\alpha}_{ij}$ and $\boldsymbol{\beta}_{ij}$ are two positive weighing variables corresponding to the distance between the $\boldsymbol{i}^{th}$ and $\boldsymbol{j}^{th}$ image representations in the shared subspace that gets computed in each iteration. The mentioned mathematical expression (6) objectifies the idea of bringing similar images closer and dispersing dissimilar images apart. If the total number of modalities (feature-sets) is $\boldsymbol{\eta}$, we formulate our problem of joint supervised matrix factorization as follows,

$$\begin{aligned} \min_{P_1, \cdots P_\eta, H} \{ & \sum_{k=1}^{\eta} \|V_k - P_k H\|_F^2 + \lambda_1 (\sum_{k=1}^{\eta} \|P_k\|_F^2 + \|H\|_F^2) \\ & + \lambda_2 (\sum_{i,j \in S} \alpha_{ij} \|h_i - h_j\|^2 - \sum_{i,j \in D} \beta_{ij} \|h_i - h_j\|^2)\} \end{aligned} \tag{7}$$

We use two control parameters $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$. Parameter $\boldsymbol{\lambda}_1$ controls the trade-off between the loss minimization and the regularization functions for all the variable terms and parameter $\boldsymbol{\lambda}_2$ controls the trade-off between the loss minimization and the integration of label information. Figure 1 shows the description of our approach (where $\boldsymbol{\eta}$=5).

3.1 Optimization

In order to solve the problem (7), we apply Alternating Optimization (AO) technique along with gradient descent to model additive updates. Each iteration comprises of two steps. In first step, we update the basis matrices $\mathbf{P}_k$'s for each modality. In the second step, we find the positive real values of $\boldsymbol{\alpha}_{ij}$'s and $\boldsymbol{\beta}_{ij}$'s by solving (6) as Linear Programming (LP) problem and then update coefficient matrix **H** (also called the shared subspace). We rewrite equation (7) in trace form as follows,

$$f = \sum_{k=1}^{\eta}\{Tr(V_k V_k^T) - 2Tr(V_k H^T P_k^T) + Tr(P_k H H^T P_k^T)\}$$
$$+ \lambda_1(\sum_{k=1}^{\eta}\{Tr(P_k P_k^T)\} + Tr(HH^T))$$
$$+ \lambda_2(2Tr(H(E-A)H^T) - 2Tr(H(F-B)H^T)) \quad (8)$$

Here, **A** ∈ $\mathbf{R}^{m\times m}$+ contains positive entries $\boldsymbol{\alpha}_{ij}$'s at the corresponding ***(i,j)***th location if that ***(i,j)***th pair of images belong to **S**, or zero otherwise. Similarly, **B** ∈ $\mathbf{R}^{m\times m}$+ contains positive entries $\boldsymbol{\beta}_{ij}$'s at the corresponding ***(i,j)***th location if that ***(i,j)***th pair of images belong to **D** or zero otherwise. **E** ∈ $\mathbf{R}^{m\times m}$+ is a diagonal matrix, whose entries are the column sums of **A**, i.e. $\mathbf{E}_{jj} = \sum_i \mathbf{A}_{ij}$, and similarly **F** ∈ $\mathbf{R}^{m\times m\,+}$ is a diagonal matrix, whose entries are the column sums of **B**, i.e. $\mathbf{F}_{jj} = \sum_i \mathbf{B}_{ij}$. ($\mathbf{X}^T$ denotes transpose of matrix **X**).

Consider the equation (8) represented by the function ***f***, which we differentiate with respect to each $\mathbf{P}_k$ for ***k*** = 1,⋯ ,***η***,

$$\frac{\partial f}{\partial P_k} = \left[-2V_k H^T + 2P_k H H^T + 2\lambda_1 P_k\right] \quad (9)$$

We use gradient descent to devise the additive updates of basis matrices $\mathbf{P}_k$ for ***k*** = 1,⋯ ,***η***, as:

$$P_k^{t+1} = P_k^t + \gamma\left[2(V_k - P_k^t H^t){H^t}^T - 2\lambda_1 P_k^t\right] \quad (10)$$

where ***γ*** is step size of the updates for each modality and t is the iteration step. To update matrix H we differentiate equation (8) with respect to **H** and obtain the following,

$$\frac{\partial f}{\partial H} = -2\{\sum_{k=1}^{\eta} P_k^T(V_k - P_k H) - \lambda_1 H +$$
$$\lambda_2(H(F-B) - H(E-A))\}$$

and the respective additive update rule for coefficient matrix **H** as,

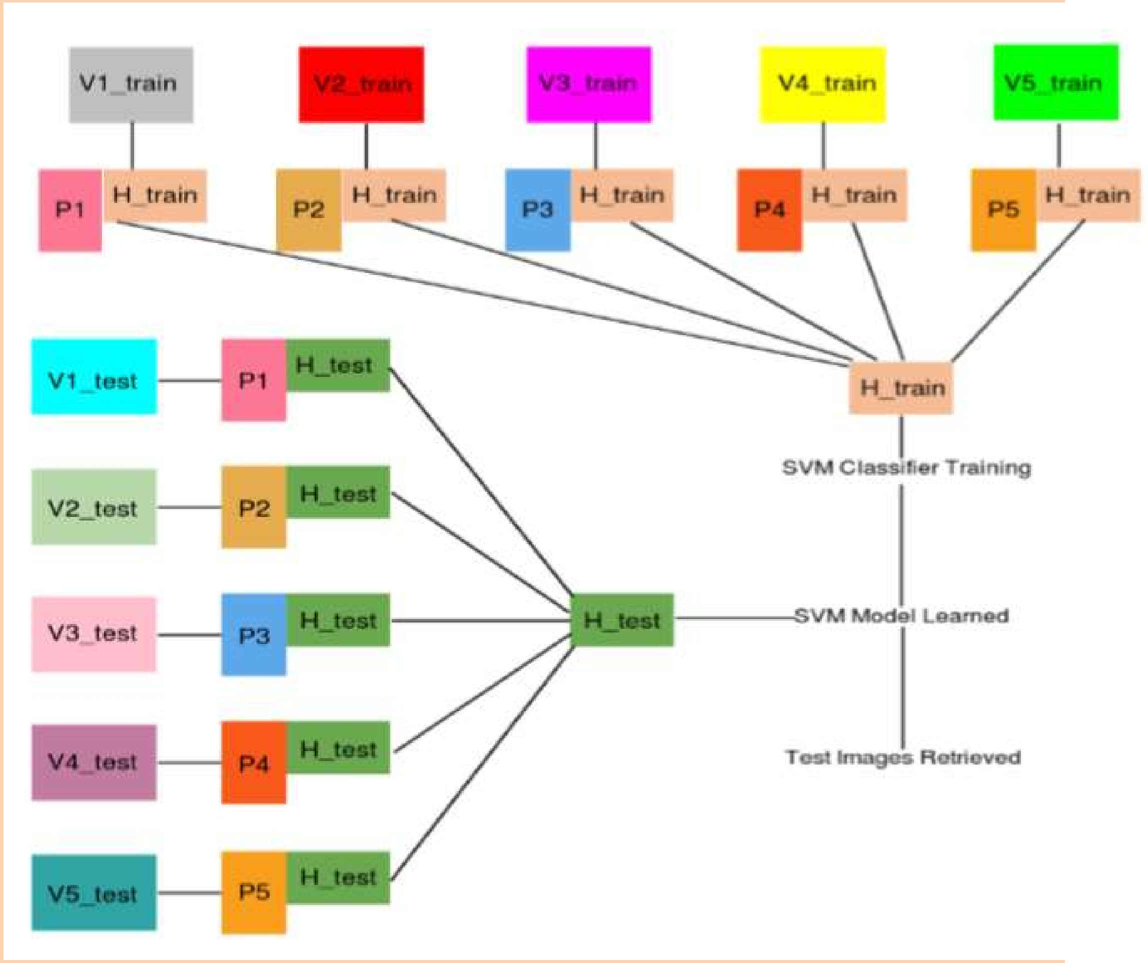


Figure 1: This figure represents our framework to obtain the informative latent subspace. $\boldsymbol{V}_{i\ train}$ *and* $\boldsymbol{V}_{i\ test}$ *are the training and test feature-sets of images respectively for* $\boldsymbol{i} = \{1,\cdots,5\}$*.* $\boldsymbol{V}_{i\ train}$ *is factorized as* $\boldsymbol{P}_i \times \boldsymbol{H}_{train}$*. The unified representation* $\boldsymbol{H}_{train}$ *is shared by all the training feature-sets. The informed basis matrices* $\boldsymbol{P}_i$ *obtained after factorization of* $\boldsymbol{V}_{i\ train}$ *are used to learn an informed* $\boldsymbol{H}_{test}$ *from* $\boldsymbol{V}_{i\ test}$ *(Notice the same colors of basis matrices). This ensures supervision in the fusion of test feature-sets also.* $\boldsymbol{H}_{train}$ *and* $\boldsymbol{H}_{test}$ *are used for training classifiers and obtaining test accuracies respectively. (Image is best viewed in color.)*

$$H^{t+1} = H^t + 2\gamma\{\sum_{k=1}^{\eta} P_k^{t^T}(V_k - P_k^t H^t) - \lambda_1 H^t + \lambda_2(H^t(F-B) - H^t(E-A))\} \quad (11)$$

where γ is the step size for additive updates. To find the values of α_{ij}'s and β_{ij}'s, we solve the (6) as linear programming problem as follows,

$$\min_{\alpha_{ij},\beta_{ij}} \Big(\sum_{i,j\in S} \alpha_{ij}\|h_i - h_j\|^2 - \sum_{i,j\in D} \beta_{ij}\|h_i - h_j\|^2\Big)$$
$$\text{such that } \alpha_{ij} > 0, \beta_{ij} > 0$$
$$\text{and } \sum \beta_{ij} - \sum \alpha_{ij} > 0$$
$$\text{and } \sum \alpha_{ij} < a, \sum \beta_{ij} < b \qquad (12)$$

where $\|h_i - h_j\|^2 = (h_i - h_j)^T(h_i - h_j)$ and values of ***a*** and ***b*** are found experimentally. These values of $\boldsymbol{\alpha}_{ij}$'s and $\boldsymbol{\beta}_{ij}$'s help to associate the label information with the shared subspace **H** in each iteration. After performing the updates (10) and (11) during training process we obtain the basis matrices (named as) $\mathbf{P}^{train}_{k}$ and the coefficient matrix (named as) $\mathbf{H}^{train}_{sup}$ which we call the informed latent subspace. Next in testing phase, we use $\mathbf{P}^{train}_{k}$ to obtain $\mathbf{H}^{test}_{sup}$ by solving the following minimization problem,

$$\min_{H^{test}} \{\sum_{k=1}^{\eta} \|V_k - P_k^{train} H^{test}\|_F^2 + \lambda_1 \|H^{test}\|_F^2\} \qquad (13)$$

Equation (13) can be written in trace form as follows,

$$f = \sum_{k=1}^{\eta} \{Tr(V_k V_k^T) - 2Tr(V_k H^{test^T} P_k^{train^T}) + Tr(P_k^{train} H^{test} H^{test^T} P_k^{train^T})\} + \lambda_1 Tr(H^{test} H^{test^T}) \qquad (14)$$

Differentiating equation (14) with respect to $\mathbf{H}_{test}$ and by applying gradient descent we obtain the following update rule,

$$H^{test^{t+1}} = H^{test^t} + \gamma[\sum_{k=1}^{\eta} P_k^{train^T}(V_k - P_k^{train} H^{test^t}) - \lambda_1 H^{test^t}] \qquad (15)$$

where $\boldsymbol{\gamma}$ is the step size for the additive update. We name our proposed method as **Joint Supervised Matrix Factorization (JSMF)**. In the next section the experimental set-up and the results are discussed in detail.

4. EXPERIMENTS AND RESULTS ANALYSIS

Experiments are conducted on two publicly available standard image datasets provided at NUS-WIDE [6], namely NUS-WIDE Object and NUS-WIDE Scene. Both the datasets have partitioned training and test sets with five different pre-computed low-level visual feature-sets for each set (training set and test set), namely **64-D** color histogram, **144-D** color auto-correlogram, **73-D** edge direction histogram, **128-D** wavelet texture and **225-D** block-wise color moments in each. Thus according to datasets, we consider $\boldsymbol{n}$ = 5 number of modals. We perform our experiments for different scales of training and test images and report results for different subspace learning and data fusion methods.

4.1 Pre-processing Step

In NUS-WIDE Object dataset, no inconsistency was found, all images in the training and test set modals were correctly matched to their corresponding labels. However some inconsistency in the labels of the image instances was observed in NUS-WIDE Scene training and test set modals. Some images were categorized in more than one categories, for which we chose any one category at random as a corrective measure. For example, in NUS-WIDE Scene dataset an image across different modals simultaneously belonged to "sky" and "sunset" categories, similarly "ocean", "beaches", and "plants" had some image instances in common. Next, some images in training set and test set of NUS-WIDE Scene dataset were found unlabeled which we removed from both the sets. Table 1 gives the description of the number of training and test images and the number of categories of images in each NUS-WIDE dataset.

4.2 Experimental Set-up

We set the environment for experimentation as CentOS 7, 48 GB RAM with 12 core 64 bit Intel Xeon 2.20 GHz processors. The five different views of images are represented as $\mathbf{V}_1,\cdots,\mathbf{V}_5$ in our model. Each of them is of dimension $\boldsymbol{m}_i\times\boldsymbol{n}$ where $\boldsymbol{n}$ is the number of images and $\boldsymbol{m}_i$ is the dimension of the $\mathbf{V}^{th}{}_i$ low-level visual feature-set. Inspired by the work of [5] an experimental framework was set up, details of which can be found in our

Table 1: Description of the Datasets

Dataset	Training Images	Test Images	Classes
Object	17,928	12,072	31
Scene	17,463 (After pre-processing: 16,586)	17,463 (After pre-processing: 16,640)	33

previous work [10] where the process of learning the shared latent subspace was unsupervised. We call this framework as Joint Unsupervised Matrix Factorization (JUMF) where the latent subspace **H** (for training and test process) was obtained by extending the iterative additive updates (3) to the five mentioned modals. As explained in previous section, in JSMF we aim to learn an informed latent subspace **H** (shared by all the five modals) via supervised matrix factorization process, i.e., $\mathbf{V}_1 = \mathbf{P}_1 \times \mathbf{H}, \cdots, \mathbf{V}_5 = \mathbf{P}_5 \times \mathbf{H}$, where $\mathbf{P}_1, \cdots, \mathbf{P}_5$ are $\boldsymbol{m}_i \times \boldsymbol{d}$ sized basis matrices and **H** is the $\boldsymbol{d} \times \boldsymbol{n}$ sized shared coefficient matrix. Parameters involved in the JUMF and JSMF model were tuned as follows: $\boldsymbol{d} \in [50,100,200]$, tolerance (as convergence criteria) = 1e-6, $\boldsymbol{\gamma}$, $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ were empirically set to 3e-4, 1e-2 and 1e-2 respectively, $\boldsymbol{C}_{\mathrm{SVM}} \in [1e2,1e3,1e4,1e5]$, $\boldsymbol{\gamma}_{\mathrm{SVM}} \in [1e\text{-}1,1e\text{-}2,1e\text{-}3,1e\text{-}4]$ in SVM and $\boldsymbol{k} = 10$ in kNN while training both classifiers. We divided the training and test set in ratio 0.6 and 0.4 approximately for different scales $\mathbf{N} \in (1600, 5000, 7500, 10000, 15000)$ images for both datasets. Thus for every round of experiment on the selected scale **N**, we select the first $0.6 \times \mathbf{N}$ and $0.4 \times \mathbf{N}$ images from each training and test modal respectively.

As explained in the previous section, in order to learn $\boldsymbol{\alpha}_{ij}$'s and $\boldsymbol{\beta}_{ij}$'s that integrate the label information in our subspace learning technique we solve equation (6) as an LP problem. However, instead of finding individual values of $\boldsymbol{\alpha}_{ij}$'s and $\boldsymbol{\beta}_{ij}$'s for each pair of similar and dissimilar images ($\boldsymbol{i}^{th}$ and $\boldsymbol{j}^{th}$ image) respectively, we employ an approximation of those to lower down the computational complexity. We approximate $\boldsymbol{\alpha}_{ij}$'s and $\boldsymbol{\beta}_{ij}$'s as $\boldsymbol{\alpha}_{cicj}$ and $\boldsymbol{\beta}_{cicj}$ respectively (where $\boldsymbol{i}^{th}$ and $\boldsymbol{j}^{th}$ image belong to class $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ respectively). For more clarity we explain the approximation procedure with the following example.

4.2.1 Example

As shown in *Figure 2*, consider three image categories **A**, **B** and **C** (where $\boldsymbol{n}_A$, $\boldsymbol{n}_B$ and $\boldsymbol{n}_C$ be the number of images in classes **A**, **B** and **C** respectively) with images $\mathbf{A}_1$,$\mathbf{A}_2$ and $\mathbf{A}_3$ in category **A**, $\mathbf{B}_1$ and $\mathbf{B}_2$ in category **B** and $\mathbf{C}_1$ and $\mathbf{C}_2$ in category **C**. The total number of similar and dissimilar pairs of images are $7 \times 7 = 49$ for 7 images. We set the approximated value $\boldsymbol{\alpha}_{cicj}$'s as $\boldsymbol{\alpha}_a = \max\{\boldsymbol{a}_{ij} : \forall\, \boldsymbol{i},\boldsymbol{j} \in \boldsymbol{n}_A\}$, $\boldsymbol{\alpha}_b = \max\{\boldsymbol{b}_{ij} : \forall\, \boldsymbol{i},\boldsymbol{j} \in \boldsymbol{n}_B\}$ and $\boldsymbol{\alpha}_c = \max\{\boldsymbol{c}_{ij} : \forall\, \boldsymbol{i},\boldsymbol{j} \in \boldsymbol{n}_C\}$, for the similar image pairs and similarly set $\boldsymbol{\beta}_{cicj}$'s as $\boldsymbol{\beta}_{ab} = \min\{\boldsymbol{a}_i\boldsymbol{b}_j : \forall\, \boldsymbol{i} \in \boldsymbol{n}_A, \forall\, \boldsymbol{j} \in \boldsymbol{n}_B\}$, $\boldsymbol{\beta}_{ac} = \min\{\boldsymbol{a}_i\boldsymbol{c}_j : \forall\, \boldsymbol{i} \in \boldsymbol{n}_A, \forall\, \boldsymbol{j} \in \boldsymbol{n}_C\}$, $\boldsymbol{\beta}_{ba} = \min\{\boldsymbol{b}_i\boldsymbol{a}_j : \forall\, \boldsymbol{i} \in \boldsymbol{n}_B, \forall\, \boldsymbol{j} \in \boldsymbol{n}_A\}$, $\boldsymbol{\beta}_{bc} = \min\{\boldsymbol{b}_i\boldsymbol{c}_j : \forall\, \boldsymbol{i} \in \boldsymbol{n}_B, \forall\, \boldsymbol{j} \in \boldsymbol{n}_C\}$, $\boldsymbol{\beta}_{ca} = \min\{\boldsymbol{c}_i\boldsymbol{a}_j : \forall\, \boldsymbol{i} \in \boldsymbol{n}_C, \forall\, \boldsymbol{j} \in \boldsymbol{n}_A\}$, $\boldsymbol{\beta}_{cb} = \min\{\boldsymbol{c}_i\boldsymbol{b}_j : \forall\, \boldsymbol{i} \in \boldsymbol{n}_C, \forall\, \boldsymbol{j} \in \boldsymbol{n}_B\}$ for dissimilar image pairs. Since images $\mathbf{A}_1$, $\mathbf{A}_2$ and $\mathbf{A}_3$ are similar to each other, instead of finding 9 $\boldsymbol{\alpha}_{ij}$ values we approximate them with only one $\boldsymbol{\alpha}_a$. Thus each category has one $\boldsymbol{\alpha}_{cicj}$ value associated with it (where $\boldsymbol{c}_i = \boldsymbol{c}_j$).

Similarly, for dissimilar pairs of images belonging to classes A and B, instead of finding 6 $\boldsymbol{\beta}_{ij}$ values, we approximate them with only $\boldsymbol{\beta}_{ab}$. Thus, each dissimilar image category pair has one $\boldsymbol{\beta}_{cicj}$ value (where $\boldsymbol{c}_i \mathrel{!=} \boldsymbol{c}_j$). A method for this approximation is explained in **4.2.2 Experiment Procedure**. Nevertheless it is important to note the reduction in number of variables for which LP problem (12) is solved, thus lowering down the computational complexity.

4.2.2 Experiment Procedure

Consider $\mathbf{H}^{train}_{unsup}$ as the trained coefficient matrix obtained after JUMF process. We pre-compute set $\mathbf{S}'$ and $\mathbf{D}'$ for each class $\boldsymbol{c}_i$ and each pair of dissimilar classes ($\boldsymbol{c}_i$, $\boldsymbol{c}_j$) respectively
using the $\mathbf{H}^{train}_{unsup}$ matrix for $\mathbf{N}^{train} \in$ (1000, 2988, 4482, 5976, 8964) training images, where set $\mathbf{S}'$ consists of image pairs ($\boldsymbol{i}, \boldsymbol{j}$) $\in$ $\mathbf{S}$ (where $\boldsymbol{c}_i = \boldsymbol{c}_j$) that are farthest from each other in class $\boldsymbol{c}_i$ and set $\mathbf{D}'$ consists of image pairs ($\boldsymbol{i}, \boldsymbol{j}$) $\in$ D (where $\boldsymbol{c}_i$!= $\boldsymbol{c}_j$) that are closest to each other in dissimilar class pair ($\boldsymbol{c}_i$,$\boldsymbol{c}_j$). Thus we have $\mathbf{T}$ image pairs in $\mathbf{S}'$, one for each class and $\mathbf{Z} = \mathbf{T}^2 - \mathbf{T}$ image pairs in $\mathbf{D}'$, one for each dissimilar class pair where $\mathbf{T}$ is the total number of classes in a dataset. In each iteration we find $\boldsymbol{\alpha}_{cicj}$ and $\boldsymbol{\beta}_{cicj}$ for those precomputed image pairs in set $\mathbf{S}'$ and $\mathbf{D}'$ respectively. Following mathematical expression objectifies the approximation concept of weights,

$$\begin{aligned} \min_{\alpha_{c_ic_j},\beta_{c_ic_j}} &\Big(\sum_{i,j\in S'} \alpha_{c_ic_j} \|h_i - h_j\|^2 - \sum_{i,j\in D'} \beta_{c_ic_j} \|h_i - h_j\|^2 \Big) \\ &\text{such that } \alpha_{c_ic_j} > 0, \beta_{c_ic_j} > 0 \\ &\text{and } \sum \beta_{c_ic_j} - \sum \alpha_{c_ic_j} > 0 \\ &\text{and } \sum \alpha_{c_ic_j} < a', \sum \beta_{c_ic_j} < b' \end{aligned} \tag{16}$$

which we solve as LP problem using a python library *cvxopt* (values of $\boldsymbol{a}'$ and $\boldsymbol{b}'$ are found experimentally). After obtaining the values of $\boldsymbol{\alpha}_{cicj}$ and $\boldsymbol{\beta}_{cicj}$ in each iteration we map those values back to the weights $\boldsymbol{\alpha}_{ij}$ and $\boldsymbol{\beta}_{ij}$, thereby constructing **F**, **B**, **E** and **A** matrices. Then executing the additive update rules (10) and (11) until convergence we obtain the basis matrices $\mathbf{P}^{train}_k$ (for $\boldsymbol{k} = 1,\cdots,\boldsymbol{\eta}$) and the sharedcoefficient matrix $\mathbf{H}^{train}_{sup}$. Next we train the support vector machine (SVM) with RBF kernel and kNN ($\boldsymbol{k}$ = 10) classifiers using the obtained informed latent subspace ($\mathbf{H}^{train}_{sup}$). In testing phase by executing the update rule (15) we obtain $\mathbf{H}^{test}_{sup}$ which we consider to be an intelligent latent subspace due to informed basis matrices $\mathbf{P}^{train}_k$'s. The whole procedure is explained in ***Algorithm 1***. Note that, since the modal spaces are same for both training and test sets we can use the informed basis matrices obtained in training phase to produce the informed Htest sup in testing phase. We obtain the image classification results by passing $\mathbf{H}^{test}_{sup}$ to the trained classifiers and compare those with the results obtained by different subspace learning and data fusion methods.

4.3 Result Analysis

Tables 2, 3, 4 and *5* show the comparative study of our proposed JSMF algorithm with other approaches of subspace learning and data fusion. FC, DLF, JSMF, GCCA, RGCCA and JUMF stand for Feature Concatenation, Decision Level Fusion, Joint Supervised Matrix Factorization, Generalized Canonical Correlation Analysis, Regularized Generalized Canonical Correlation Analysis and Joint Unsupervised Matrix Factorization respectively.

Algorithm 1 JSMF

```
Input: $V_i\_train$, $V_i\_test$ matrices for $i = 1, \cdots, \eta$, $H^{train}_{unsup}$, tolerance $tol$
Output: $H^{test}_{sup}$ and $H^{train}_{sup}$
Precomputation: Use $H^{train}_{unsup}$ to compute $S'$ and $D'$.
Training:
repeat
    Update $P^t_k$ using (10).
    Compute $A, B, E, F$:
     – Solve LP (16) to get $\alpha_{c_i c_j}$ and $\beta_{c_i c_j}$
     – Map $\alpha_{c_i c_j}$ and $\beta_{c_i c_j}$ to $\alpha_{ij}$'s and $\beta_{ij}$'s respectively
     – Assign $\alpha_{ij}$ and $\beta_{ij}$ to $(ij)^{\text{th}}$ position of matrices A
       and B respectively
     – Compute column sum matrices E and F
    Update $H^t$ using (11)
until ($H^{t+1} - H^t = tol$ and $P^{t+1}_k - P^t_k = tol$)
$H^{train}_{sup} = H^t$ and $P^{train}_k = P^t_k$
Testing:
repeat
    Update $H^{test}$ using (15)
until ($H^{test\,t+1} - H^{test\,t} = tol$)
$H^{test}_{sup} = H^{test}$
```

Table 2: SVM - % accuracies on NUS-WIDE Object

Data Fusion Methods	Number of Images				
	1600	5000	7500	10000	15000
FC	23.0	**33.54**	37.0	38.94	40.93
DLF	15.17	24.05	28.86	30.81	32.28
JSMF (d=50)	20.5	29.0	34.8	37.8	**41.3**
JSMF (d=100)	23.0	31.6	**37.0**	39.2	41.0
JSMF (d=200)	**23.5**	29.8	36.3	**39.8**	40.8
GCCA	2.5	13.07	16.33	12.35	12.97
RGCCA	5.83	5.71	9.37	19.50	9.64
JUMF (d=50)	13.0	19.2	26.5	30.2	35.0

Table 3: kNN - % accuracies on NUS-WIDE Object

Data Fusion Methods	Number of Images				
	1600	5000	7500	10000	15000
FC	15.5	22.41	26.93	27.95	29.73
DLF	12.33	20.92	27.10	28.50	29.97
JSMF (d=50)	**16.83**	**24.40**	**27.53**	**29.05**	**31.7**
JSMF (d=100)	14.83	21.67	24.98	26.71	29.10
JSMF (d=200)	14.83	18.59	21.70	23.01	25.72
GCCA	10.33	7.5	8.5	6.5	6.6
RGCCA	3.67	3.72	9.7	16.47	6.99
JUMF (d=50)	13.33	23.06	26.44	27.63	30.16

The figures reported in tables are maximum classification accuracies obtained over values of C_{SVM} and $\gamma_{SV\,M}$ mentioned earlier in **4.2 Experimental Set-up** subsection and k = 10 for kNN. For FC experiment we simply concatenated all the features sets end to end thereby forming a data matrixb with m×634 dimension where m is the number of images. But due to high dimensionality of data it led to increased training and test time of SVM and kNN classifiers. In DLF experiment we trained individual classifiers on each of the five modals and did a maximum majority voting among the output classes obtained in the test phase. As explained in **4.2 Experimental Set-up** subsection we report results for various values of d for JSMF. For GCCA and RGCCA we used the regCCA() library function implemented in R [18], setting the regularization parameter equal to 0.5 for RGCCA experiment. In JUMF experiment the unsupervised latent subspace H was obtained by extending the iterative additive updates (3) to the five modals. However, the results obtained in JUMF for d=50 were significantly less than JSMF and thus not promising, so further JUMF experiments with other two values of d were omitted. From the tables it can be easily inferred that our proposed model JSMF (with due note of JUMF) performs well with the low dimensional representation of images with the value of d equal to 50 or 100, thereby indicating a successful data fusion and subspace learning.

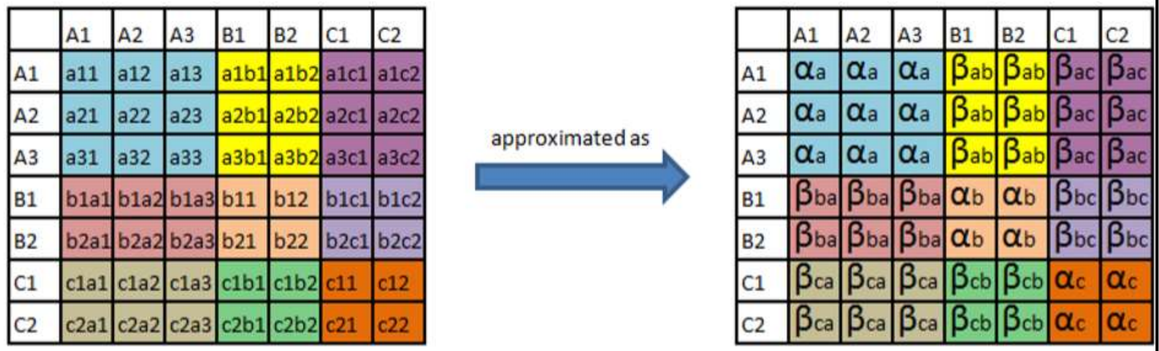

A1 A2 A3 B1 B2 C1 C2
A1 a11 a12 a13 a1b1 a1b2 a1c1 a1c2
A2 a21 a22 a23 a2b1 a2b2 a2c1 a2c2
A3 a31 a32 a33 a3b1 a3b2 a3c1 a3c2
B1 b1a1 b1a2 b1a3 b11 b12 b1c1 b1c2
B2 b2a1 b2a2 b2a3 b21 b22 b2c1 b2c2
C1 c1a1 c1a2 c1a3 c1b1 c1b2 c11 c12
C2 c2a1 c2a2 c2a3 c2b1 c2b2 c21 c22
approximated as
A1 A2 A3 B1 B2 C1 C2
A1 αa αa αa βab βab βac βac
A2 αa αa αa βab βab βac βac
A3 αa αa αa βab βab βac βac
B1 βba βba βba αb αb βbc βbc
B2 βba βba βba αb αb βbc βbc
C1 βca βca βca βcb βcb αc αc
C2 βca βca βca βcb βcb αc αc

Figure 2: The left figure shows the individual α_{ij}, (a_{ij}, b_{ij}, c_{ij}) and β_{ij}, $(a_i b_j, a_i c_j, \cdots, c_i b_j)$ for each similar and dissimilar pair of images. The right figure shows the approximation of those individual α_{ij}'s and β_{ij}'s as approximated $\alpha_{c_i c_j}$ where $(c_i = c_j)$ and $\beta_{c_i c_j}$ where $(c_i \,!= c_j)$ respectively. (Image is best viewed in color.)

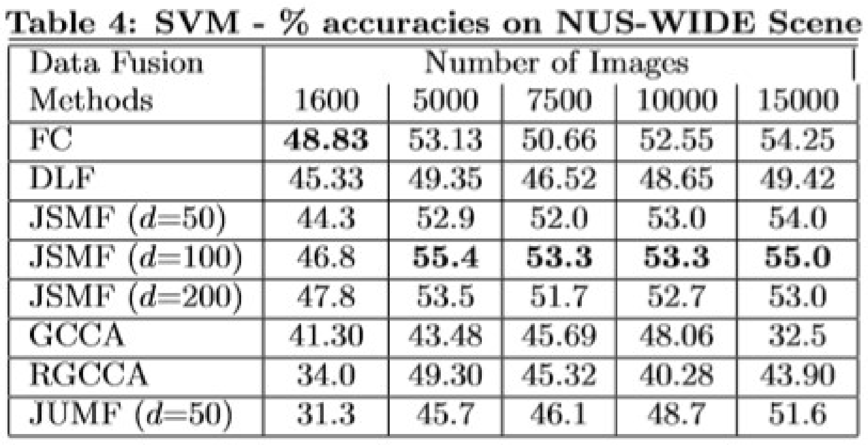


Table 4: SVM - % accuracies on NUS-WIDE Scene

Data Fusion Methods	Number of Images				
	1600	5000	7500	10000	15000
FC	**48.83**	53.13	50.66	52.55	54.25
DLF	45.33	49.35	46.52	48.65	49.42
JSMF (d=50)	44.3	52.9	52.0	53.0	54.0
JSMF (d=100)	46.8	**55.4**	**53.3**	**53.3**	**55.0**
JSMF (d=200)	47.8	53.5	51.7	52.7	53.0
GCCA	41.30	43.48	45.69	48.06	32.5
RGCCA	34.0	49.30	45.32	40.28	43.90
JUMF (d=50)	31.3	45.7	46.1	48.7	51.6

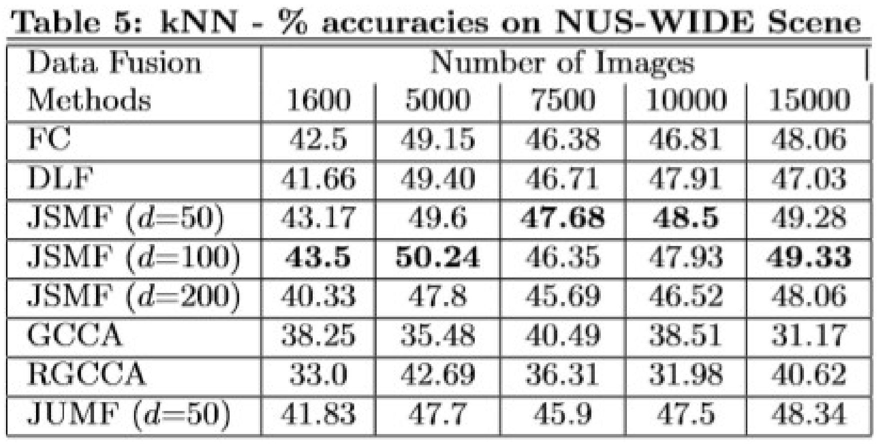


Table 5: kNN - % accuracies on NUS-WIDE Scene

Data Fusion Methods	Number of Images				
	1600	5000	7500	10000	15000
FC	42.5	49.15	46.38	46.81	48.06
DLF	41.66	49.40	46.71	47.91	47.03
JSMF (d=50)	43.17	49.6	**47.68**	**48.5**	49.28
JSMF (d=100)	**43.5**	**50.24**	46.35	47.93	**49.33**
JSMF (d=200)	40.33	47.8	45.69	46.52	48.06
GCCA	38.25	35.48	40.49	38.51	31.17
RGCCA	33.0	42.69	36.31	31.98	40.62
JUMF (d=50)	41.83	47.7	45.9	47.5	48.34

However, with the value of ***d*** set to 200, the performance of JSMF model deteriorates which we can attribute to possible feature redundancy due to high dimension. Other subspace learning methods were not compared due to their inherent limitation to two modals only. One important point to note with NUS-WIDE Object dataset is that subspace learning techniques GCCA and RGCCA perform very poorly; clearly indicating that subspace learning on this dataset is a tough task. Perhaps this explains the distributed maximum classification accuracies obtained over the various versions of JSMF on this dataset in the case of SVM classification (*Table 2*). In the case of kNN classification (*Table 3*), JSMF with ***d*** = 50 clearly outperforms all other approaches. For NUS-WIDE Scene dataset too, in SVM classification (*Table 4*) JSMF with ***d*** = 100 outperforms other approaches. Similar correspondence is observed in kNN classification (*Table 5*) of NUS-WIDE Scene dataset. Thus we successfully show that our proposed JSMF method performs comparable or better than existing state-of-the-art methods for subspace learning and data/information fusion.

5. CONCLUSION

We have presented a new label-driven non-convex shared subspace learning technique that can be extended to large-scale datasets due to additive iterative updates. We integrate the label information with subspace learning process that makes the retrieval accuracy better. As of now, a number of convex optimization techniques are available for convex relaxation that perform better and faster. With the aim of applying convex relaxation on JSMF we also aim to evaluate our technique on cross-modal datasets. Also, several speeding-up embeddings like Stochastic Gradient Descent etc. may be applied to improve the convergence runtime. Finally, we aim to analyze the algorithm for error bounds.

6. ACKNOWLEDGMENT

The authors would like to acknowledge Department of Computer Science & Engineering at IIT-BHU (Varanasi) for their financial support and computational resources.

C. INTERMEDIATE SUBSPACE LEARNING IN ZERO SHOT FRAMEWORK

ABSTRACT

Image classification and retrieval have been a long-standing area of active research. Classic image retrieval techniques have an inherent limitation due to the problem setting itself. Here the models need to learn from a given image database (training images) and later classify similar test images which belong to the same classes as in training set. What if we do not have image samples for few classes while training, although we would need to classify unseen images belonging to those classes when the model is put to the test. For example, the system is trained with images of dogs and cars, but it needs to accurately predict the images of horses. As a solution to this problem, the concept of Zero-Shot Learning (ZSL) was recently formulated. This paper presents a brief survey of existing approaches in ZSL as well as discusses future possibilities for which the researchers have ample opportunities, owing to the fact that this is a relatively new and active research area in image classification and retrieval.

Keywords - Image Classification, Zero-Shot Learning, Intermediate Semantic Space

1. INTRODUCTION

Image classification has been a long-standing area of research in machine learning. Content-Based Image Retrieval (CBIR) techniques have dominated this domain, which has attracted a lot of bright minds, with significant research done and still continuing. More specifically, in CBIR one tries to learn the mapping $\boldsymbol{f : X \rightarrow Y}$, where $\boldsymbol{f}$ is the function which maps images in $\mathbf{X}$ to labels or classes in $\mathbf{Y}$. One inherent limitation of state-of-the-art approaches in CBIR is the sharing of same label space by training and test data. In other words, for training dataset $\mathbf{X}_{\text{train}}$ and test dataset $\mathbf{X}_{\text{test}}$, the labels set $\mathbf{Y}_{\text{train}}$ and $\mathbf{Y}_{\text{test}}$ respectively share same label space. Thus for each test class image, we have few corresponding training images belonging to the same class.

But let us bring into picture the real world facts, there are 30,000 classes of objects which can be identified by human beings. Is it possible to gain training instances for each class to train our classifier model, which we expect to classify the unseen objects accurately at test time? Certainly not. Collecting a large number of training instances of each class is infeasible owing to high cost of collecting training instances, extensive human effort in labelling them etc. .This limitation is overcome by a newly emerging approach, called, **Zero Shot Learning** (**ZSL**) for classifying objects of whose class no training instances are present in training dataset. That is, the labels set $\mathbf{Y}_{\text{train}}$ and $\mathbf{Y}_{\text{test}}$ are disjoint, they do not share the same label space.

In fact, the name Zero Shot Learning is derived from the presence of zero instances of test classes in training set, thus zero-data or zero-shot. More specifically, say for example we have a training set, with images from classes horse, chair and car and we need to classify images belonging to classes elephant and bed during the testing phase. Before proceeding further in ZSL, it would be just to bring into notice that Fei Fei et al. [1] had already proposed one shot learning, where for their models only one image of test classes during training would suffice. Lampert et al. [9] first introduced the problem of ZSL, where they addressed it in the most intuitive way in their seminal paper.

For the sake of originality, we re-state the ZSL problem setting they [9] had set up. They created a benchmark dataset Animal with Attributes, (AwA) which has 30,475 animal images each belonging to one of the 50 animal classes. They define attribute vectors of each class, where each attribute vector is an 85-dimensional binary vector. Physical features and characteristics, for example, is the animal furry? Does the animal have a tail?, does it live in water? etc. comprise the attributes. The set of 50 classes is divided into two sets, training set comprising 40 classes and test set comprising rest 10 classes thereby making training and test set disjoint. Task is to classify the test images correctly in one of those 10 test classes.

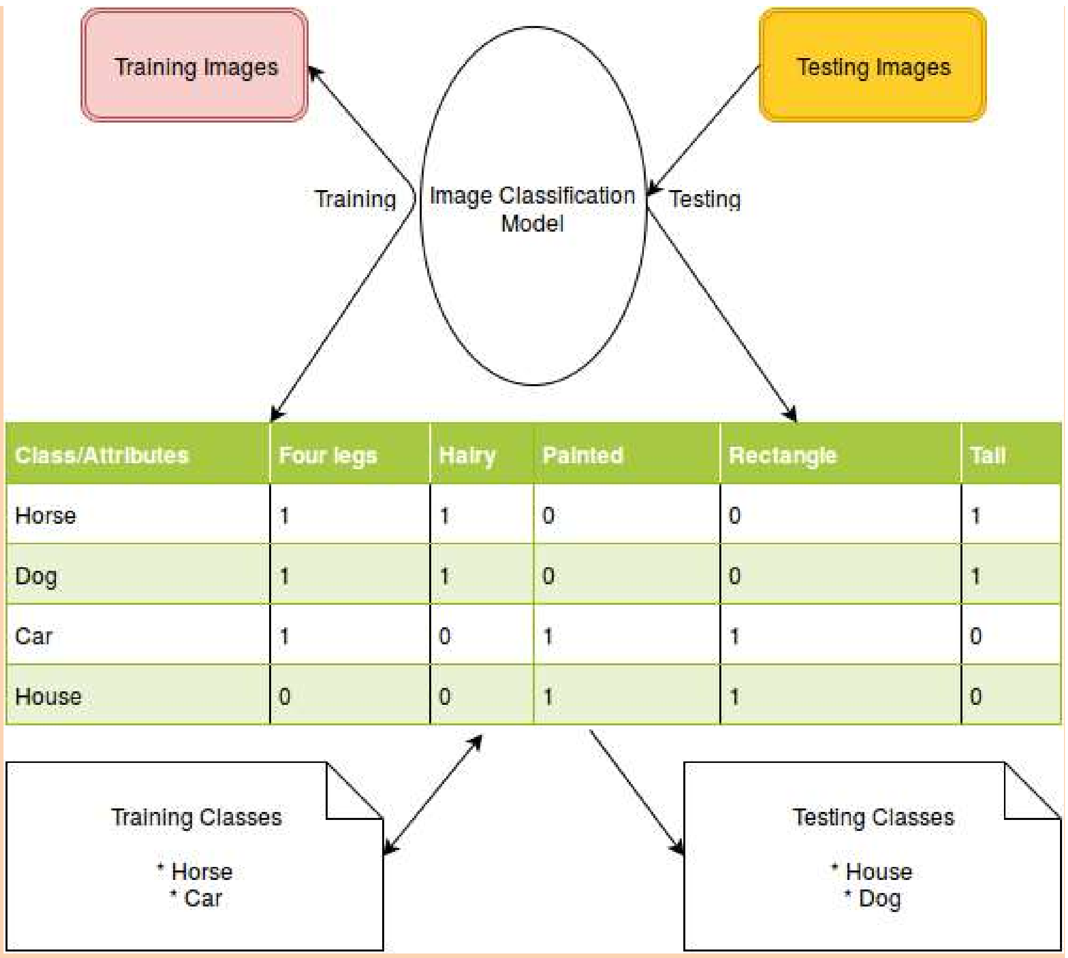
Since the inception of this interesting problem, many notable research scientists have contributed to its solutions. Reviewing the multiple approaches so far, we found that they rely on information transfer through an intermediate semantic space which they construct either with attributes or word vectors. It has been almost ten years since the seminal paper by Lampert et al. [9] and till date there has been no survey paper which briefly describes the ZSL problem statement and approaches to it. We present the first survey paper for ZSL where we discuss above-mentioned two approaches ***Zero Shot Learning through Attributes*** and ***Zero Shot Learning through Word Vectors***. Section 2 is dedicated to approaches in ZSL which use attributes, and the next section 3 discusses methods which utilize word-vectors information. Section 4 presents a quick discussion on shortcomings in existing approaches, future possible research directions and real life applications of ZSL. We conclude our work in section 5.

2. ZERO-SHOT LEARNING THROUGH ATTRIBUTES

Attributes, as mentioned earlier are the high-level characteristics of objects easily identifiable by human beings, which in the case of animals, say for elephant are: it is big, it has a trunk, it has a small tail, it is not furry, for lion, it has attributes like: it is big, it has no trunk, it has a large tail, it is furry and so on for different classes. Before we proceed further explaining the papers which adopted attributes for ZSL let us intuitively understand how they can be helpful in confronting this problem of recognizing unseen class images, whose no training samples are present.

Imagine a scenario of a father teaching his child to recognize animals in a national park. *Hey son, remember what I taught you, if you happen to find a four-legged animal with a small tail, black in color, big in size but no furry body, large ear lobes, and a trunk then it is an elephant. Okay?* Yes Dad, but I see there a big animal yellow-brown in color with furs, it has a tail too, but no trunk. What can that be? *Oh son, that can be either a lion or a tiger. Do you see any stripes over its body?* No Dad. *Good, that is a lion then.* Wow... a lion. Hey Dad look over there, I see a big animal, with furs all over its big yellow-brown body, it has a tail too but it has stripes all over its body. It is a tiger...Right? *Yes Son! It is a tiger.*

Above short tale sums up the attribute based approach to ZSL. The child was taught to recognize characteristic properties (attributes) of animals which helped him to identify them. Next, he was also able to learn subtle differences among tigers and lions based on the texture of the body i.e. striped. Can we build a system to recognize attributes given an image of an animal, and then due to the relation of attributes and class of that animal, can we predict unseen class animals? Almost all of works under this category follow an approach which revolves around learning to recognize attributes and then exploiting the relation between attributes and classes. Mostly the models learn to recognize attributes from images, and when presented with unseen class image, identifies the attributes in it and then classifies it. This is how information is transferred from training samples to test samples by constructing an attribute based intermediate semantic space.



Class/Attributes	Four legs	Hairy	Painted	Rectangle	Tail
Horse	1	1	0	0	1
Dog	1	1	0	0	1
Car	1	0	1	1	0
House	0	0	1	1	0

Figure 3: Zero-Shot Learning through Attributes framework.

Below are the various approaches categorized as Batch Learning where all input data is provided before executing the model, and Online Learning where input data is fed in incremental order while the model executes.
****
****

2A: BATCH LEARNING

2A1: Attribute Based ZSL: Lampert et al. [10]

Authors herein first proceed with definition of attributes as a high level property of an object, easily identifiable by human beings. So considering the animal categories for example elephant, it has attributes like: it is big, it has a trunk, it has small tail, it is not furry, for lion, it has attributes like: it is big, it has no trunk, it has large tail, it is furry and so on for different classes. They define an attribute space A, which is a binary attribute representation of human identified attributes of animal classes. They propose that they can learn a non-trivial classifier $\boldsymbol{\alpha} : \mathbf{X} \rightarrow \mathbf{Y}_{\text{test}}$ by transferring information between $\mathbf{Y}_{\text{train}}$ and $\mathbf{Y}_{\text{test}}$ through $\mathbf{A}$, thus naming it Attribute based Intermediate Semantic Space.

They come up with two methods to learn attribute based classification namely **Direct Attribute Prediction (DAP)** and **Indirect Attribute Prediction (IAP)**. As the name suggests, DAP model learns to predict attributes from training images and training labels, which later is used to predict attributes for test images thus labeling them with the help of attribute based intermediate semantic space. IAP, on the other hand learns a classifier for each training class $\mathbf{Y}_{\text{train}}$ and use the predictions during testing to infer the attributes from which labeling of test classes is done. Authors support both methods with plausible statements. They state that direct learning of attributes as in DAP, is favorable in situations where training and test classes are treated equally. Thus when inferring test class labels, the decision is based on the model learned on attributes. However, in case of IAP, since model which learn attributes are learned from classes and not from images can lead to a bias if $\mathbf{Y}_{\text{train}}$ and $\mathbf{Y}_{\text{test}}$ are not disjoint. It can also be argued that learning to infer attributes from class labels instead of raw images can make the system more robust by learning only sensible attribute combinations based on classes. Following probabilistic mathematical interpretations clarifies the two concepts.

In nutshell, authors realize both the techniques DAP and IAP by combination of a supervised classifier or regressor for image-attribute or image-class prediction, with parameter free inference to transfer information through attribute layer.

2A1.1: Direct Attribute Prediction

Authors start by learning direct image-attribute probabilistic classifiers for each attribute $\boldsymbol{a}_{\text{m}}$ .For the sake of better interpretation let us construct a matrix with rows corresponding to each class and columns corresponding to each attribute inferred by mechanical Turks. The matrix has binary values. Thus all images belonging to training class $\boldsymbol{y} \in \mathbf{Y}_{\text{train}}$ share same attribute vector $\boldsymbol{a}^{\text{y}}$, where $\boldsymbol{a}^{\text{y}}_{\text{m}}$ denote the $\boldsymbol{m}^{\text{th}}$ attribute of class ***y***. Authors have used 85 attributes in all for 50 classes. Thus the size of the matrix can be considered at 50 × 85.

They learn probabilistic classifiers $p(\boldsymbol{a}_m \mid \boldsymbol{x})$ i.e. the conditional probability of attribute $\boldsymbol{a}_m$ given class ***x***, from which they model the attribute vector as $p(a|x) = \prod_{m=1}^{M} p(a_m|x)$. Once this attribute vector classifier is learned, during test time they apply Bayesian rule to learn

$$p(z|a) = p(a|z)\frac{p(z)}{p(a^z)}$$

where they assume $p(a|z) = [[a = a^z]]$, such that [[P]] = 1 if P is true else 0 otherwise.

Combining both equations, they obtain probability of class ***z*** given the image is ***x*** as

$$p(z|x) = \sum_{a \in 0,1^M} p(z|a)p(a|x) = \frac{p(z)}{p(a^z)} \prod_{m=1}^{M} p(a_m^z|x) \tag{1}$$

Note that intuitively we needed to first of all predict the attribute vector of test image ***x*** (corresponding term $p(\boldsymbol{a}|\boldsymbol{x})$: image-attribute) and then we also need to find what is the probability that the test image ***x*** belong to class ***z*** (corresponding term $p(\mathbf{z}|\mathbf{a})$: attribute-image), thus multiplying both the terms to get the overall probability. This method of DAP is also the current state-of-the-art benchmark for batch learning [8].

2A1.2 Indirect Attribute Prediction

For IAP, authors modify only the image-attribute term, as here they need to do indirect attribute prediction, instead of being direct. As written earlier, for IAP authors learn multiclass classifier $p(\boldsymbol{y}_k|\boldsymbol{x})$ i.e. the probability of class begin $\boldsymbol{y}_k$ given the image ***x***, where $\boldsymbol{y}_k \in \mathbf{Y}_{train}$. Once again authors assume $p(\boldsymbol{a}_m|\boldsymbol{y})=[[\boldsymbol{a}_m=\boldsymbol{a}^y{}_m]]$. Combining both terms, they obtain following equation

$$p(a_m|x) = \sum_{k=1}^{K} p(a_m|y_k)p(y_k|x) \tag{2}$$

Next its easy to obtain $p(\boldsymbol{a}|\boldsymbol{x})$ as shown earlier in DAP method and then proceed in the same way as in DAP.

2A2: Embarrassingly simple approach to ZSL : Paredes et. al. | 2015 [14]

The authors here again visit the concept of intermediate semantic space representation in terms of attribute feature space, where their two linear layers network model learns the relationship from image visual features to attribute features and from attribute features to class labels. They simplify the problem of ZSL into two sub-stages, first one being training stage where attributes are learned from images, and the second one being inference stage where class labels are inferred from learned attributes. Similar to the work of Lampert et al. [10]. they design an attribute matrix $\mathbf{S} \in [0,1]^{a\times z}$ where ***a*** stands for the dimension of attribute vector for ***z*** training classes. The way they represent their training images and training labels, it can be extended to multi-class classification also. $\mathbf{X}_{\text{train}} \in \mathbf{R}^{d\times m}$ where ***d*** is the dimensionality of visual features and ***m*** is the number of training samples and $\mathbf{Y}_{\text{train}} \in [-1,1]^{m\times z}$ which is a matrix denoting the class of an image to which it belongs. Thus for a single class label image, only one entry in its corresponding row is positive and others negative, but for a multi-class label image, relevant columns (i.e. classes) are positive. They design a loss function of form below

$$minimize L(X_{tr}^T VS, Y) + \Omega(V) \quad (3)$$

where $\mathbf{V} \in \mathbf{R}^{d\times a}$ is the matrix to be learned, ***L(.)*** is a loss function, and ***Ω(.)*** is the regularizer. After **V** has been learned, the unseen class of instance ***x*** can be predicted from the following equation

$$argmax_i x^T VS' \quad (4)$$

where $\mathbf{S}'$ is the semantic feature matrix of the new set of unseen class images. The loss function not only learns how to recognize the attributes but also learns the importance of them for different classes. ***L(.)*** can be generalized to different loss functions like hinge loss, squared error loss etc. . Setting the regularizer of the form as $\boldsymbol{\Omega(B)=\Psi(B^TB)}$ and using Representer Theorem, following kernel version of the problem is formulated

$$minimize_{A\in R^{m\times a}} L(KAS, Y) + \Psi(S^T A^T KAS) \quad (5)$$

also the following regularizer is formulated

$$\Omega(V;S,X) = \gamma\|VS\|_{Fro}^2 + \lambda\|X^T V\|_{Fro}^2 + \beta\|V\|_{Fro}^2 \quad (6)$$

where ***γ,λ,β*** are the regularizer parameters and $\|.\|_{Fro}^2$ is the Frobenius norm. Interestingly they claim that solution to their approach is just one line of code which they formulate as below

$$V = (XX^T + \gamma I)^{-1} XYS^T (SS^T + \lambda I)^{-1} \quad (7)$$

2B. ONLINE LEARNING

2B1: Fast Online Incremental Transfer Learning for Unseen Object Classification Using Self-Organizing Incremental Neural Networks: Kawewong et. al. | 2011 [8]

The authors were the first ones to address the problem of online zero-shot learning where they confronted the problem of learning incrementally from the data as it gets available to them and proposed the online incremental attribute learning and knowledge transferring system. In a previous work [6], the authors had already devised an incremental learning method Self-Organizing Incremental Neural Network (SOINN), which they use in combination with SVM, naming it Alt-SOINN-SVM. This Alt-SOINN-SVM is used for transfer learning. Authors first proceed with two independent incremental learning strategies, Neural Networks, and SVM, and later they combine SOINN (a neural network architecture) with SVM, for achieving improved results.

2B1.1 Self-Organizing Incremental Neural Network (SOINN)

Let us begin with a short and precise introduction to SOINN. It is a self-organizing incremental unsupervised online clustering and learning method, which models a neural network to represent data distribution with nodes representing sample data and edges representing relationships among them. Beginning with an empty set of nodes, for two consecutive input samples, it creates two nodes. Then for the next coming inputs, it finds two nearest nodes (say $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$) and calculates the distance between $\boldsymbol{s}_1$ and $\boldsymbol{x}$ and between $\boldsymbol{s}_2$ and $\boldsymbol{x}$. If both of these two distance measures are smaller than some threshold, then the input $\boldsymbol{x}$ belong to the inner cluster. In this case, some updates are done in the weight vector of node $\boldsymbol{s}_1$ and either an edge is created between nodes $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ or the existing edge weight is reset. If the distances are not less than the threshold, then a new node is created and added to the network.Thus it limits the memory requirement for SOINN, as it need not remember/represent each incoming input data. Also, as analogous to the human brain, certain extraneous information pertaining to noise is lost if the edges and nodes corresponding to that info is not updated for a specified period of time, thus making SOINN less prone to noise in input data.

Authors use the same notations as in [10], with $(\boldsymbol{x}_1,\boldsymbol{y}_1),\cdots,(\boldsymbol{x}_n,\boldsymbol{y}_n) \in \mathbf{X}_{\text{train}}\times\mathbf{Y}_{\text{train}}$ denoting training pairs (samples $\boldsymbol{x}_i$, and label $\boldsymbol{y}_i$), $\mathbf{Z}$ denoting test classes disjoint from $\mathbf{Y}_{\text{train}}$ and $\boldsymbol{a} \in \mathbf{A}$ denoting a binary attribute vector, where $\boldsymbol{a}^y$ denotes attribute vector of a class $\boldsymbol{y} \in \mathbf{Y}_{\text{train}}$ and $\boldsymbol{a}^z$ denotes attribute vector of a class $\boldsymbol{z} \in \mathbf{Z}$. We outline the Alt-SOINN-SVM method for which we describe first Alt-SOINN and Alt-SGD-SVM methods as prerequisites.

2B1.2 Alt-SOINN

Beginning with the definition of SOINN [6], it is an online unsupervised classification learning mechanism, which performs like an online incremental clustering tool [8]. By its definition, let us consider SOINN to be a binary classifier of individual attributes a^{y}_{i} in an attribute vector a^{y}. Similar to the work of Lampert et al. [10], first of all attribute learning is done and then transfer learning is done to classify unseen classes. Authors define **M** SOINNs for **M**-dimensional attribute vector a such that each SOINN infers the presence or absence of ai in an image. They divide each SOINN into two parts, namely positive and negative part where they contain growing clusters for images containing that particular attribute ai and for images not containing it respectively. So if the **Q** features are used then, there is a need for **Q**×**M** SOINN binary classifiers. During the training period, training visual feature vector of one feature is inputted incrementally into corresponding **M** SOINNs for that visual feature (authors have used 6 visual features of images). Using the attribute labels for 50 classes in [10], knowledge about positive and negative images is obtained, and thus each SOINN is trained for corresponding attributes a_i. During test phase, on an input of test sample's one visual feature to every corresponding trained SOINN, a positive and negative set of k nearest neighbors nodes in positive and negative part is obtained represented as $S^{+}_{i} = \{s^{+}_{i,1}, \cdots, s^{+}_{i,k}\}$ and $S^{-}_{i} = \{s^{-}_{i,1}, \cdots, s^{-}_{i,k}\}$ respectively. Here $s^{+}_{i,m}(m \in [1,k])$ denote the mth positive nearest node in ith SOINN, note the i varies from [1,**M**] for **M** SOINNs corresponding to one feature, similarly for $s^{-}_{i,m}$. Let d^{+}_{i} stand for the average Euclidean distance in between the test sample and k nodes in S^{+}_{i} , similarly for d^{-}_{i} . Thus we can obtain a vector $d^{+} = (d^{+}_{1}, \cdots, d^{+}_{M})$ and $d^{-} = (d^{-}_{1}, \cdots, d^{-}_{M}) \in \mathbf{R}^{M}$ for each corresponding test sample image.

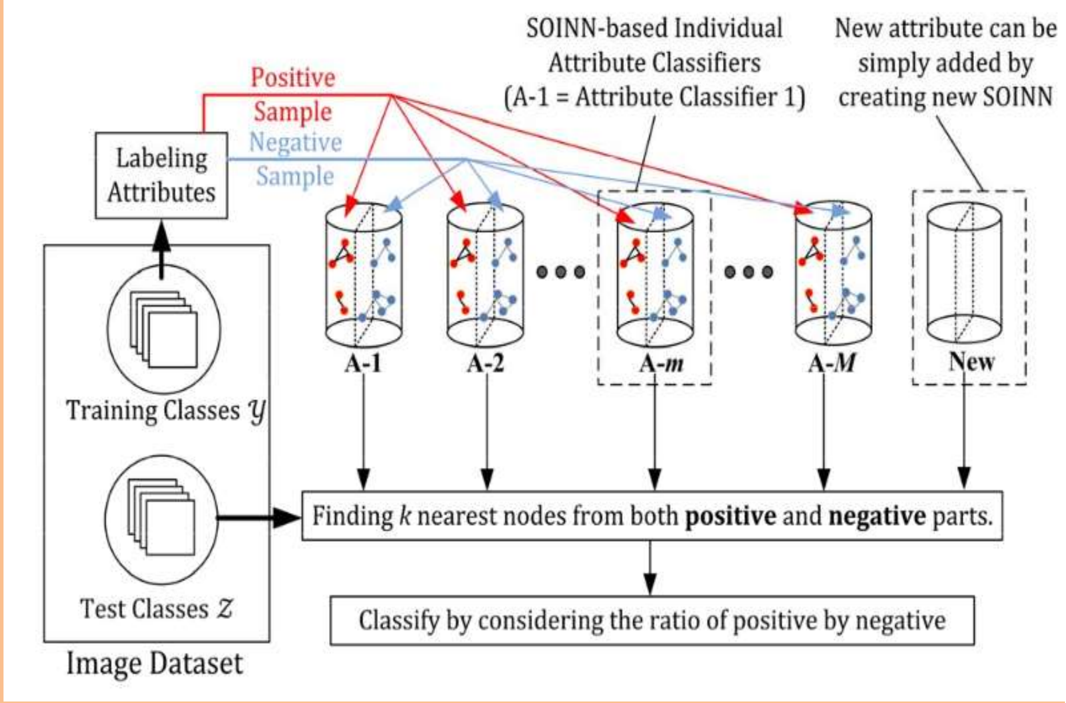

SOINN-based Individual Attribute Classifiers (A-1 = Attribute Classifier 1)
New attribute can be simply added by creating new SOINN
Positive Sample
Negative Sample
Labeling Attributes
Training Classes $\mathcal{Y}$
Test Classes $\mathcal{Z}$
Image Dataset
A-1
A-2
A-m
A-M
New
Finding k nearest nodes from both positive and negative parts.
Classify by considering the ratio of positive by negative

Figure 0.4: Illustration of Alt-SOINN Courtesy: [8]

Authors formulate and make use of the following equation to classify a test image visual feature x to an unseen class z_c.

$$c = argmax_{l=1,\cdots,L}(\sum_{q=1}^{Q} a_{z_l}^{q}) \quad (8)$$

where α^q_{zl} is the similarity score of class z_l for qth feature, where it is obtained by

$$\alpha_{z_l} = \sum_{i=1}^{M} \frac{d_i^+[[a_i^{z_l} = 1]] + d_i^-[[a_i^{z_l} = 0]]}{d_i^+ + d_i^-} \quad (9)$$

where [[**P**]] denote Iverson's bracket such that it evaluates to 1 if **P** is true else 0.

2B1.3 Alt-SGD-SVM

SVM is inherently designed for batch learning and classification and not for online incremental learning. However, by the usage of stochastic gradient descent (SGD), SVM can be made an online incremental learning tool. By definition, SGD enables SVM to update its classification parameters on the basis of few randomly selected training samples. In this case, one randomly selected sample or one incoming data sample can update the parameters of SVM, thus making it online. Similar to the previous work by Lampert et al. [10] and Alt-SOINN, Alt-SGD-SVM first of all estimates $p(\boldsymbol{a}_i|\boldsymbol{x})$ for a given sample $\boldsymbol{x}$ and then predicts the unseen class z of $\boldsymbol{x}$ as $p(z|\boldsymbol{x})$ through the following equation

$$p(z|x) = \frac{p(z)}{p(a^z)} \prod_{i=1}^{M} p(a_i^z|x) \quad (10)$$

Authors model Alt-SGD-SVM as a linear SVM with hinge loss objective function $l(\boldsymbol{u}) = max\{0, 1-\boldsymbol{u}\}$, with $\boldsymbol{u}=\boldsymbol{a}(\boldsymbol{wx}+\boldsymbol{b})$ where $\boldsymbol{w}$ is the weight vector to be updated in every iteration. Since the main emphasis is on the discussion of Alt-SOINN-SVM, further details about Alt-SGD-SVM are omitted, and interested authors are referred to the paper for more information.

2B1.4: Alt-SOINN-SVM

This method is the combination of the best properties of SOINN and SVM and thus uses them to perform different tasks. Apparently, SOINN is used for online incremental clustering and learning due to its ability to reduce noise as it updates the nodes iteratively with each new training sample and retain only those who represent more relevant information, and SVM is used for classification on the arrival of a test sample due to its better discriminating ability. Authors state that Alt-SOINN-SVM is fully online incremental in its learning part but offline in its classification part.

They are able to use linear SVM because SOINN clusters the data making them more linearly separated, thus classification time is also reduced making it sufficient for online classification. As described in Alt-SOINN section, here also individual SOINN classifiers for attributes $\boldsymbol{a}_i$ is learned for each feature, thus amounting to total $\mathbf{Q}\times\mathbf{M}$ SOINN. Remember that each SOINN used to have positive nodes and negative nodes which denoted information about samples containing the attribute $\boldsymbol{a}_i$ and samples not containing it. So during the training phase of linear SVM, those two sets of positive and negative nodes are used as samples for positive and negative classes. Note that, this way we obtain $\mathbf{Q}\times\mathbf{M}$ linear binary classifier SVMs. Thus this combination of SOINN with SVM can be considered the state-of-the-art method for online ZSL problem.

2B2: Online Incremental Attribute-based ZSL | Kankuekul et. al. | 2012 [7]

This work can be considered an extension and improvement to the previous online approach by the same authors. In addition to this, once again they derive inspiration from the work of Lampert et. al. and more explicitly exploit the Indirect Attribute Prediction (IAP) model proposed in[10].Their previous model Alt-SOINN-SVM, the combination of SOINN and SVM was based on Direct Attribute Prediction and suffered from lower accuracy and slower computation time compared to this model. Henceforth data notations are consistent with immediately previous section's.

Recollect that DAP model would first learn a direct classifier for predicting attributes from training data and then during test phase would predict the M attributes and then with the help of attribute matrix, unseen class of input test image would be determined. On the contrary, IAP model first of all learns a classifier for training classes $\boldsymbol{y}_k \in \mathbf{Y}_{train}$ instead of attributes. During the test phase, for an input test image $\boldsymbol{x} \in \mathbf{X}_{test}$ attributes are inferred from the learned class classifiers, and then unseen classes are predicted from the learned attributes. Intuitively one can think that DAP would perform better than IAP, and it sure does, but authors use IAP for obvious reasons written below.

• Simpler implementation of IAP than DAP

• Low computational cost and Flexibility in attribute prediction, of IAP than DAP

DAP has to learn more number of classifiers (pertaining to attribute classifiers) in compared to IAP which learn only limited number of object classifiers qualified by the number of training classes. Also in the case of an addition of an attribute (which can be expected more frequently compared to addition of a class in online system), DAP has to learn a new attribute classifier whereas IAP uses existing seen class classifiers. Thus we can infer that IAP has more flexibility in case of an addition of an attribute to the learning system compared to DAP, as we would be needed to train multiple new models for every addition of new attributes or for every change in labeling of attributes in DAP.

• Suitability for long-term learning

Above points can vouch for it. Mislabeling and attribute addition can occur throughout the online learning process in which case, IAP certainly has benefits over DAP.

Authors investigate and online both methods DAP and IAP for comparison purpose, by borrowing their previous online model SOINN-SVM. Contrary to the probabilistic and statistical classifiers for DAP and IAP, as proposed by Lampert et al. [10] authors here use SOINNs in place of them to accomplish the classification task.

2B2.1: DAP-SOINN-SVM (DAP-SS)

As goes for DAP, we need to learn attribute classifiers first. Here also authors learn SOINN attribute classifiers but slightly in unconventional way compared to previous work. Previously each SOINN classifier had positive and negative parts symbolizing presence and absence of attributes in image samples respectively. But here for each attribute ai two individual SOINNs are modeled, one capturing the info about positive samples and the other capturing info about negative ones. Thus overall for **M** attributes, 2×**M** SOINNs are required. So for an attribute $\boldsymbol{a}_i$, an image sample characterizing it is fed into positive SOINN and the sample which does not have it is fed to a negative SOINN. Keep in mind that SOINNs are just online data representation and clustering framework, we still need SVM for each attributes to learn attribute classifiers. So output nodes from two SOINNs for each attribute are used to train a linear SVM thus getting overall **M** attribute classifiers for **M** attributes.

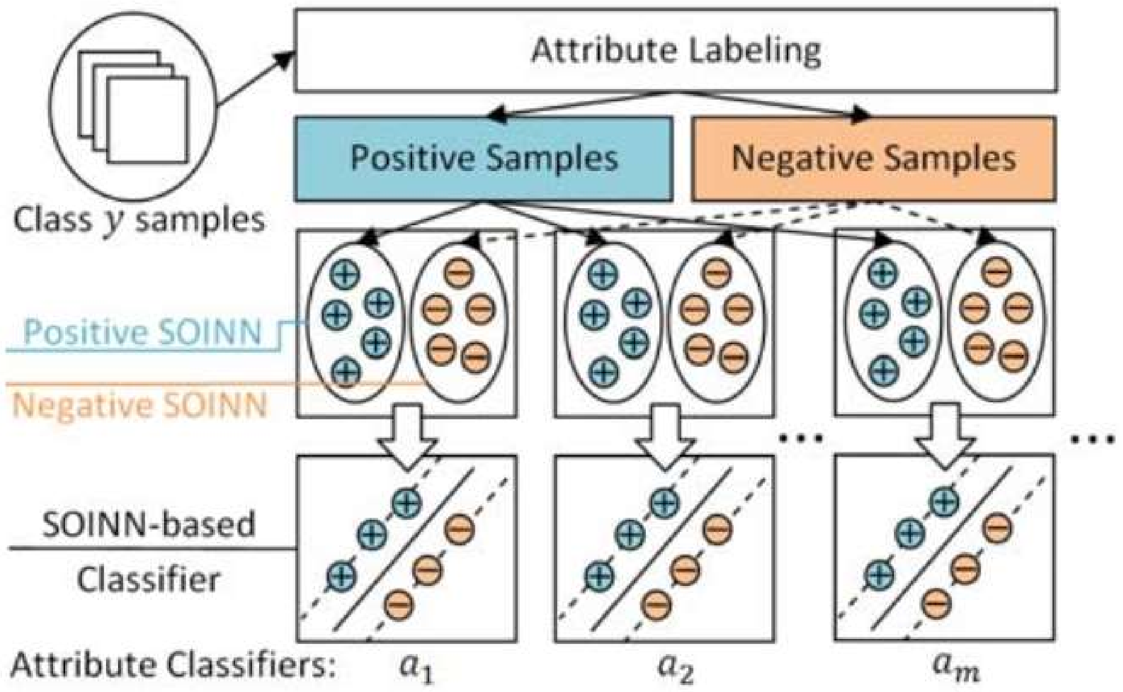


Figure 0.5: Illustration of DAP-SOINN-SVM Courtesy: [7]

During test phase, a test image is inputted to the **M** classifiers, thereby obtaining **M** decision values $\{\boldsymbol{d}_1,\cdots,\boldsymbol{d}_M\}$, where

$$d_i = f_i(x) = w_i.x + b_i, \forall i \in [1, M] \quad (11)$$

Since this decision values only represent the presence or absence of attributes and we need the probability of an attribute $\boldsymbol{a}_i$ given an image sample $\boldsymbol{x}$, the authors formulate the following equation

$$p(a_i|x) = \eta(d_i) \quad (12)$$

where $\boldsymbol{\eta}$ is a sigmoid function

$$\eta(d_i) = \frac{1}{1 + e^{-d_i}} \quad (13)$$

B2.2 IAP-SOINN-SVM (IAP-SS)

Similarly, for IAP, we need to learn **K** object classifiers instead of attribute classifiers, so the authors train **K** SOINNs for **K** classes. They feed in the training images of each class to its corresponding SOINN classifier. Next, they train **K** SVMs by gathering result nodes from all SOINNs, with one set of nodes from **k**th SOINN being as positive samples and rest begin negative samples for the corresponding SOINN. For attribute prediction of a test images $\boldsymbol{x}$, it is inputted to every classifier, and from the obtained decision values $\{\boldsymbol{d}_1, \cdots, \boldsymbol{d}_K\}$, authors obtain the $p(\boldsymbol{y}_k|\boldsymbol{x})$ as

$$p(y_k|x) = \prod_q \eta(d_k^q) \quad (14)$$

hereby considering the output for every feature space $\boldsymbol{q} \in \mathbf{Q}$. Then they predict the attribute $\boldsymbol{a}_i$ given image $\boldsymbol{x}$ as

$$p(a_i|x) = \sum_k p(a_i|y_k)p(y_k|x) \quad (15)$$

where $p(\boldsymbol{a}_i|\boldsymbol{y}_k) = [[\boldsymbol{a}_i = \boldsymbol{a}^{yk}_i]]$ is inferred from the attribute matrix as described in Lampert et al. [10], and [[.]] is the Iverson's bracket.

2B2.3 Predicting the test class

Both the above DAP-SS and IAP-SS methods show how to predict the attribute a_i given an image x, i.e. $p(a_i|x)$ but how to predict the test class z_l for a given test image. This can be done by making use of the following equation:

$$p(z_l|x) = \sum_i p(z_l|a_i)p(a_i|x) \quad (16)$$

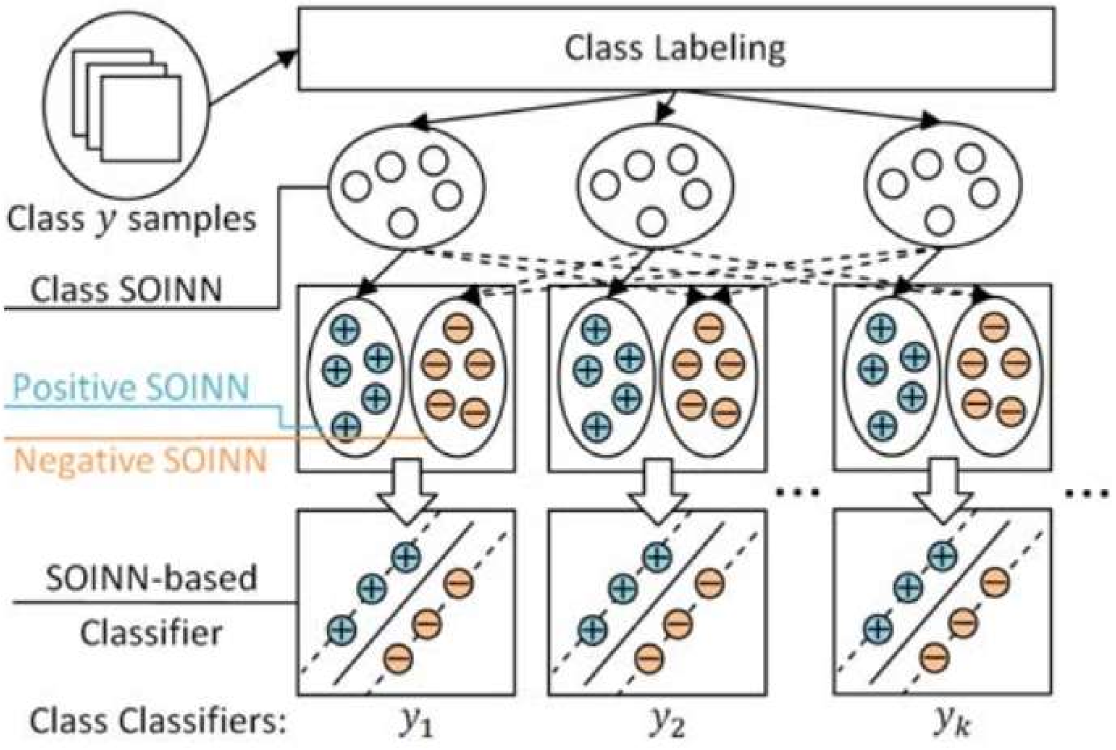


Figure 0.6: Illustration of IAP-SOINN-SVM Courtesy : [7]

We already have inferred $p(\boldsymbol{a}_i|\boldsymbol{x})$ but we need the value of $p(z_l|\boldsymbol{a}_i)$, which authors find as:

$$p(z_l|a_i) = \frac{p(a_i|z_l)p(z_l)}{p(a_i)} \qquad (17)$$

where $p(\boldsymbol{a}_i|z_l)$=[[$\boldsymbol{a}_i = \boldsymbol{a}^{z_l}_i$]] is inferred from attribute matrix and $p(z_l)= 1/\mathbf{L}$ where **L** is the number of test classes and $p(a_i) = \sum_k \frac{a_i^{y_k}}{K}$ where **K** is the number of the training class. Since we need to account for the probability of occurrence of attribute $\boldsymbol{a}_i$, we average its occurrence in all classes $\boldsymbol{y}_k$. Now we have the $p(z_l|\boldsymbol{x})$ and its final belonging class can be simply found by taking $argmax_{z_l}(p(z_l|x))$.

3. ZERO SHOT LEARNING THROUGH WORDVECTORS

3A. BATCH LEARNING

3A1: Zero-Shot Learning through Semantic Output Codes | Palatucci et al. | 2009 [13]

Palatucci et al. built a Semantic Output Code (SOC) classifier for solving the problem of decoding neural images to corresponding words thought by people from fMRI scans of their neural activity. Since the corpora of words in literature is humongous, it is not possible to learn to decode for every neural activity image mapping of words. This brings out the latent zero-shot learning problem in it. Authors begin with the definition of Semantic Feature Space **F**, which they define as a binary metric space which encodes the value of a semantic property. This semantic feature space is similar to the attribute matrix defined in previous work by Lampert et al. [10]. Next, they define a Semantic Knowledge Base **K** as a collection of pairs $(\boldsymbol{f},\boldsymbol{y})_{1:M}$ such that $\mathbf{f} \in \mathbf{F}^p$ is a point in a **p** dimensional semantic space $\mathbf{F}^p$ and $\boldsymbol{y} \in \mathbf{Y}$ is a class label from a set **Y**. Finally, they define Semantic Output Code Classifier $\mathbf{H}:\mathbf{X}^d \rightarrow \mathbf{Y}$, which maps points from some $\boldsymbol{d}$ dimensional raw-input space $\mathbf{X}^d$ to a label from a set **Y** such that **H** is the composition of two other functions, **S** and **L**, such that:

$$H = L(S(.)), S : X^d \rightarrow F^p, L : F^p \rightarrow Y \quad (18)$$

We can observe that this model, first of all maps a $\boldsymbol{d}$ dimensional input space to a $\boldsymbol{p}$ dimensional semantic feature space which is constructed from word-vectors obtained from **Google-Trillion-Word-Corpus** and then from intermediate semantic space to label space. During the training phase, along with input space $(\boldsymbol{x},\boldsymbol{y})$ where $\mathbf{x} \in \mathbf{X}^d$ and $\boldsymbol{y} \in \mathbf{Y}$, a knowledge base **K** is also provided. They learn the **S** map by building a new set of training examples $(\boldsymbol{x},\boldsymbol{f})$ with the help of **K**. Now the task remains is to learn **S** map, which authors do by using a multiple output linear regression. To get a more clear idea, using the authors' notation let $\mathbf{X} \in \mathbf{R}^{N\times d}$ be training feature matrix representation, where **N** is the number of training samples $\mathbf{X}_{train}$ and $\boldsymbol{d}$ are the dimension of each image feature, and $\mathbf{Y} \in \mathbf{R}^{N\times p}$ be the corresponding semantic feature matrix representation of **N** images, which is obtained from **K**. They learn a weight matrix $\mathbf{W} \in \mathbf{R}^{d\times p}$, by formulating following equation:

$$W = (X^T X + \lambda I)^{-1} X^T Y \quad (19)$$

such that each image of ***d*** dimensional feature vector can be mapped to ***p*** dimensional intermediate semantic feature space. Once semantic features of images are learned they use 1-nearest neighbour classifier to infer the nearest label $\boldsymbol{y} \in \mathbf{Y}_{test}$ from corresponding ***f***. No doubt, this approach seems similar to work done by Lampert et al. [10] but instead of modeling on probabilistic classifier, authors chose regressors.

3A2: ZSL through Cross-Modal Transfer | Socher et al. | 2013 [17]

As the name suggests, two different modals image and text are exploited to implement zero-shot learning problem. This research is different from previous works on account of the fact that, authors do not build an intermediate semantic space, rather than they employ data-based representations i.e. projections from one metric space to another, here from image metric space to text metric space. They build semantic word vectors from text and try to understand how the objects look like by observing the distribution of words in the metric space. Various methods like skip-gram approach, bag of words etc. are applied to capture semantics of words and represent them as word vectors.

Beginning with the construction of ***d*** dimensional semantic word vector space **F**, authors learn projections from image visual feature space to word vector space. They train a neural network for learning the projections of training images $\boldsymbol{x}_i \in \mathbf{X}_{train}$ mapped to corresponding class label word vectors $\boldsymbol{w}_y \in \mathbf{Y}_{train}$ by minimizing the following objective function

$$J(\Theta) = \sum_{y \in Y_{tr}} \sum_{x^i \in X_y} \| w_y - \theta^2 f(\theta^1 x^i) \|^2 \tag{20}$$

where $\boldsymbol{\theta}^1 \in \mathbf{R}^{h \times I}$, $\boldsymbol{\theta}^2 \in \mathbf{R}^{d \times h}$ and $\boldsymbol{f} = tanh$. $\boldsymbol{\Theta}$ is defined as $(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2)$. They train their cost function with a two-layer neural network with standard back propagation and L-BFGS. It can be easily inferred from equation that for every training class label ***y*** and thus corresponding word vector $\boldsymbol{w}_y$, they try to minimize the squared loss between the observed/actual word vector $\boldsymbol{w}_y$ and calculated ***d*** dimensional representation of image sample $\boldsymbol{x}_i$ for every image in corresponding training class $\mathbf{X}_y \in \mathbf{X}_{train}$. One another novelty in their work is that they extend their model to conditions where the test set $\mathbf{X}_{test}$ consists of images from seen classes $\mathbf{Y}_{train}$ as well as images from unseen classes $\mathbf{Y}_{test}$, where they apply two different strategies for segregating and classifying unseen class images from seen class images. Interested reader are referred to their paper for more information.

3A3: Ridge Regression, Hubness and Zero-Shot Learning | Shigeto et. al. | 2015 [16]

In addition to addressing the problem of ZSL, this paper also addresses the well-known problem of hubness [18–20] i.e. emergence of hubs in kNN search step.They mathematically prove that projection from target space (i.e. label word vector space) to source space (i.e. visual feature space) reduces the emergence of hubs and leads to efficient and more accurate kNN search in ZSL. Hubs are defined as points in a metric space which lie in the nearest neighbor search space of many mapped query points. As the title of paper suggests they use Ridge Regression to learn mapping by projecting samples from label space ($\mathbf{Y}_{\text{train}}$) to example space ($\mathbf{X}_{\text{train}}$) instead of projecting from example space to label space to avoid emergence of hubs.

3A3.1 : Emergence of Hubness

Let us first try to understand the emergence of hubness problem in a metric space. Authors show that mapped data points with ridge regression or ordinary least squares, into target space tend to be closer to origin than the target points, and since the mapped points constitute a large number of points in source space, hubs emerge near the location of origin. The phenomenon of hubness also depends on the variance of data to be projected. If the variance of two sets of data say s^2_1 and s^2_2 respectively for ***data***$_1$ and ***data***$_2$ such that $s^2_1 < s^2_2$ then the points ∈ ***data***$_1$ having shorter expectation lie closer to the origin than points ∈ ***data***$_2$, given both data are in same metric space.

Authors prove that since out of two points $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ with $\boldsymbol{y}_1$ lying closer to origin than $\boldsymbol{y}_2$ but both points equidistant from given query point $\boldsymbol{x}$, $\boldsymbol{y}_1$ is highly likely to be the nearest neighbor of $\boldsymbol{x}$, a distribution with smaller variance would be preferred for the label space $\mathbf{Y}_{\text{train}}$, thus $\boldsymbol{x}_i$ should belong to a distribution ***data***$_2$ with greater variance and $\boldsymbol{y}_i$ should belong a distribution ***data***$_1$ with smaller variance to suppress the emergence of hubs. Thus qualifying the projection of target/label space to source/image space and then performing kNN search step.

3A3.2: Modeling ZSL as Ridge Regression problem

Proceeding with authors' notations of source object metric space and target object metric space, let $\mathbf{X}_{\text{train}} = \{\boldsymbol{x}_i | i=1,\cdots,\boldsymbol{n}\}$ and $\mathbf{Y}_{\text{train}} = \{\boldsymbol{y}_i | i=1,\cdots,\boldsymbol{n}\}$ denote the training samples in source space and their label word-vectors in target space respectively, such that $\boldsymbol{y}_i$ is the label of $\boldsymbol{x}_i$ where $\mathbf{X}_{\text{train}} \in \mathbf{R}^{d\times n}$ and $\mathbf{Y}_{\text{train}} \in \mathbf{R}^{c\times n}$. They learn a projection $\boldsymbol{m}:\mathbf{R}^d \rightarrow \mathbf{R}^c$ through regression technique such that

$$M = argmin_M(\|MY_{tr} - X_{tr}\|_F^2 + \lambda\|M\|_F) \quad (21)$$

where λ is a hyper parameter and $\|.\|_F$ stands for Frobenius norm and $\mathbf{M} \in \mathbf{R}^{d\times n,}$ thereby learning the projection $\mathbf{M}$ mapping target objects $\mathbf{Y}_{\text{train}}$ to source objects $\mathbf{X}_{\text{train}}$ space.

4. DISCUSSION

We saw two main knowledge bases for information transfer, namely ***attribute based source*** and ***data-driven based representations*** i.e. word vectors. Attribute based learning focuses on learning attributes and then classifying a test instance accordingly with the help of attribute-class relation matrix. Data-driven representations rely on learning projections (read the semantic relationship between images and words) of images and labels in an intermediate semantic space or projection of images in word vector space. Although the methods presented here and other works successfully tackle it with increasing success, there are still other areas to improve the existing models. The shortcomings in the current approaches, future research directions and applications of ZSL are discussed below.

I. Issues in current approaches to ZSL

1) Domain Shift Issue: Almost all the strategies to tackle ZSL problem suffer from domain shift problem. Domain Adaptation refers to the ability of the model to execute as desired on target(test) dataset after learning from the training dataset. Recollect that source (training) and target (testing) dataset are completely different. Classification methods learned through source dataset are biased to it. Thus we can expect biased results in the case of unbalanced disjoint datasets. For domain adaptation, we need to have samples from source dataset and target dataset such that the learning is robust and there is as little error as possible in predicting samples from target dataset. Transfer learning is an inevitable part of domain adaptation where there is knowledge transfer between source dataset and target dataset, but we should keep in mind that ZSL is not a transfer learning approach although it is similar to it. A very recent work by Ji et al. [6] explores the possibility of domain adaptation to confront the domain shift issue. Other possibilities of domain adaptation can also be looked for and exploited in ZSL problem.

2) Lack of Attribute information: Almost all methods discussed here are solely based either on attribute based learning or word vectors based but not on the combination of both. Being attribute based, constant human supervision is needed to identify new attributes and thus cumbersome. It is not practical to have attributes defined for a large number of classes, thus raising the need for data-driven representations. Online methods developed so far typically use attributes, thus marking the need for online approaches which would use word vectors or a combination of both. While using word vectors in online models, one can also formulate it as a problem of NLP.

II. Other less explored approaches

1) Dictionary Learning: In dictionary learning, for a given set of instances we try to find a sparse basis matrix representation whose linear combination can span the source domain. Recollect that source and target domain are completely disjoint. However, label space can be joint. Learning a sparse basis matrix by finding relationship from the source domain to joint label space can be leveraged in ZSL setting with the disjoint target domain.

2) Neural Nets: Only a few papers [16],[2] and [3] and online methods [7] and [8] use neural networks for leveraging information from attributes and word vector spaces, and they have proven to be of superior success with greater model flexibility. From the short tale in the introductory section, it should be evident that approaches which better mimic human learning should be devised for better results. Neural Networks do this job well. Therefore current research in ZSL can focus on them too.

III. Applications of Zero-Shot Learning

1) Neural Activity Decoding: Authors in [13] experiment their ZSL model for neural activity decoding. The application sounds rightly intuitive from ZSL point of view. The task is to predict the word or object that the human is thinking about by observing the image of neural activity of the brain. Had this been a traditional image classification problem, there would have been a fixed set of words or object for which prediction could be done. But in real setting large corpora of words is present, it is not possible to train a model for each word by obtaining its visual representation (i.e. neural activity image). ZSL solves this problem in limited training resources. Their model learns the semantics of words and uses it to infer new words corresponding to unseen neural activity image.

2) Machine Vision in Robotics: Real-time robots interact with continuously changing the environment, and like humans, they need some learning mechanism to infer information about the unseen object by its visual representation. It's not possible to train robots for each object in the environment, rather than as the humans do, they should be able to learn about the properties of new objects by inferring out the attributes from its image or through textual descriptions. ZSL enables robots to accomplish this task because they learn to infer attributes to further acquire information about the object by classifying it accurately. Robust online models can be of great use here.

Other applications include in field of computer vision, drug discovery, bilingual lexicon extraction and cross-lingual information retrieval, etc. .

D. PROPOSED ALGORITHMS AND EXPERIMENTS ON ZERO-SHOT LEARNING

This section presents the algorithms proposed for Zero-Shot Learning and related experiments done along with results. The disjoint-classes visual datasets and wordvector datasets experimented to evaluate the proposed models are explained below.

1.IMAGE DATASETS

1A. Animal With Attributes (AwA) Dataset

It contains images of **30475** animals belonging to **50** animal categories. **40** animal classes are reserved for constructing training set i.e. images in training set belong to those **40** classes only (**24295** images). Images in test dataset belong to rest of the **10** classes (**6180** images). Thus the training and test classes are disjoint. They are:

Animal Types for Training: antelope, grizzly bear, killer whale, beaver, dalmatian, horse, German shepherd, blue whale, Siamese cat, skunk, mole, tiger, moose, spider monkey, elephant, gorilla, ox, fox, sheep, hamster, squirrel, rhinoceros, rabbit, bat, giraffe, wolf, chihuahua, weasel, otter, buffalo, zebra, deer, bobcat, lion, mouse, polar bear, collie, walrus, cow, dolphin.

Animal Types for Testing: chimpanzee, giant panda, leopard, persian cat, pig, hippopotamus, humpback whale, raccoon, rat, seal.

As explained earlier, we need an intermediate semantic space for information transfer. In this dataset, attribute information is given which is used as an intermediate semantic space for knowledge transfer. The attributes are high-level features which are visually noticeable as well as semantically observed.For **50** classes there are **85** attributes each, so the intermediate semantic space has dimension **50x85**, where each row is a binary feature vector corresponding to a class, where **1** represents the presence of an attribute and **0** denotes its absence.

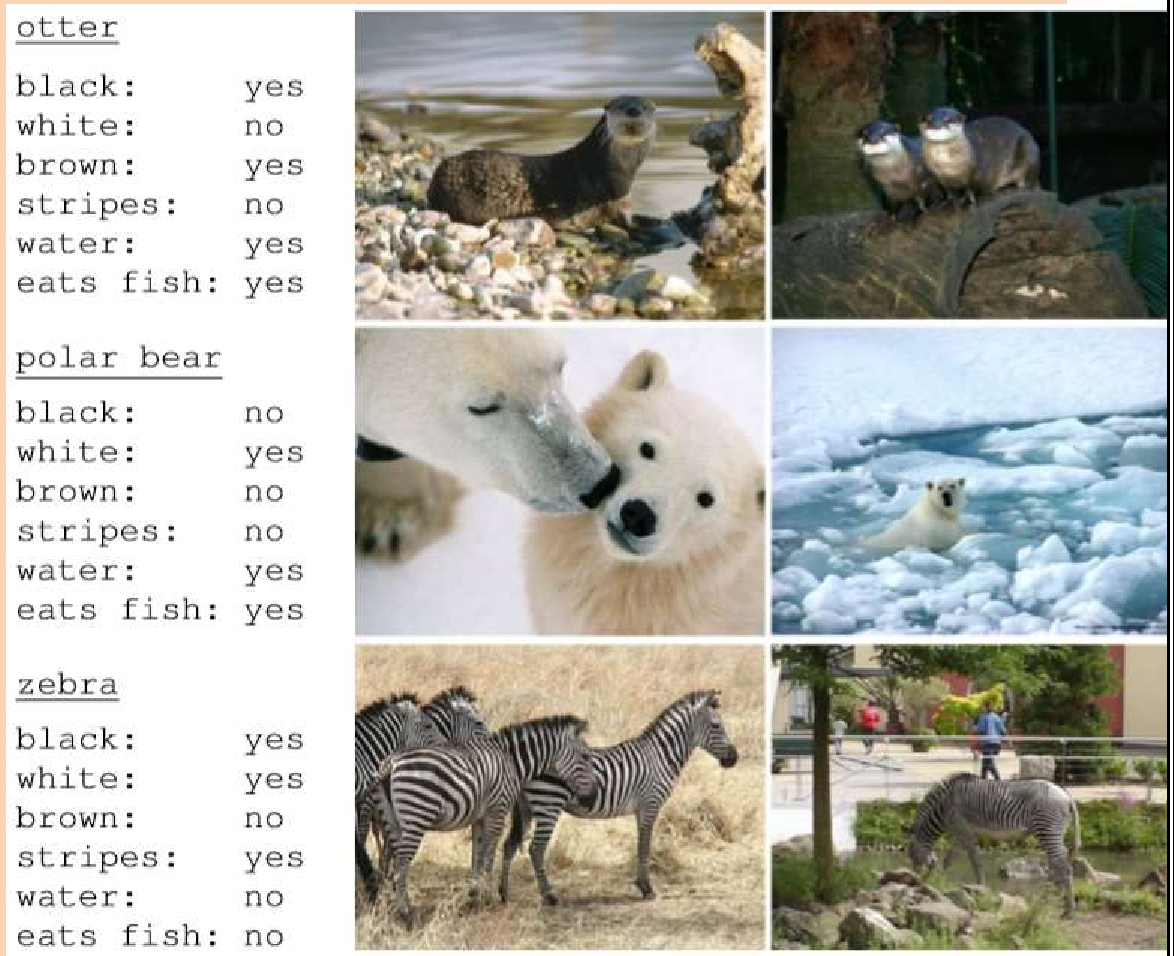


Figure: Attributes for the classes otter, polar bear, and zebra are shown, presence or absence of which is shown by yes or no. Courtesy: Lampert

1B. a-Pascal and a-Yahoo (aPaY) dataset

This image dataset is prepared from Pascal VOC 2008 dataset, where the training set comprises of **12695** images of **20** classes of a-Pascal dataset (VOC2008 training set) and test set consists of **2644** images of **12** classes of a-Yahoo dataset (VOC2008 validation set). This dataset also provides high-level binary attributes at image level (instead of class level as given in AWA dataset) for knowledge transfer. There are **64** attributes of each image. For experiment purposes, the common notion to create a class level attribute vector is to take a majority voting among attributes of images belonging to that particular class.

1C. Caltech UCSD Birds 200-2011 (CUB) dataset

This image dataset contains **200** bird categories with a total number of images being **11788**. It has **312** binary attributes per image. Similar to the above dataset, to generate a class level attribute vector, majority voting among attributes across all the images of a class was done.

2. WORDVECTOR DATASET

2A. Google News Vectors

This dataset was used to obtain word-vectors for the class labels. It is a collection of **3 Million** word-vectors with each of dimension **300**. The word-vectors are numeric real-valued vectors which obey the semantic relations between words. For example, the word-vectors for "*human*" and "*mankind*" are placed near to each other. Direction of the vector from "*man*" to "*woman*" is almost same as the direction of vector from "*king*" to "*queen*".

3. VISUAL FEATURES OF IMAGES

The visual features of images used during experiments were precomputed and provided by the authors [cite Lampert et al.]. Both shallow features (visual features hand crafted via mathematical expressions to capture visual properties of images) and deep features (visual features obtained from deep neural nets) are available, but owing to the better representation of visual information in deep features, only they (deep features) were considered for experiments. The deep features were extracted from **DeCAF** net, **GoogLeNet**, and **CNNM2K**.

4. EXPERIMENTS

Multiple algorithms were evaluated on the above datasets which are explained here as follows

> Motivation
> Algorithm
> Results
> Possible Shortcomings in Model
> Technology

4A. Formulating a Linear Projection based model

4A1. Motivation:

All the previous approaches to ZSL as explained in section C utilize only either Attribute information or Word Vector information, but do not club them together. Attributes provide direct high-level visual information of images and words provide a semantic information of images (more specifically, semantic relations between images of different entities). With the hope of leveraging information from both the attributes and word-vectors, a linear projection based model with matrix operations was formulated.

An important point to notice is that, had only been wordvectors used as semantic knowledge space for visual features projection, those projections would be biased towards training set, as only training images word vectors would have been present. But if attribute information of all the classes (training and test classes, with due note that training set still consist of images from disjoint training classes) is used, perhaps a better unbiased visual projections could be learned by exploiting informed transfer learning ("informed" because the learned projection matrices will have certain information of test classes, if not test classes images).

4A2. Algorithm

Following squared loss, function was formulated which was minimized with the help of gradient descent. As shown through the loss function, the images are projected to wordvector space, and while learning the projection matrix, information from attributes is also leveraged.

We differentiate the loss function with respect to **W** and **A**, and obtain the following additive updates which are run until convergence criteria are reached.

Later after the projections of test images are learned in wordvector space, kNN is applied to obtain the nearest words to the predicted wordvectors.

Input: Deep Visual Features, Attribute Matrix (training + test classes), Word Vector Matrix
Output: **M** matrix $\in \mathbf{R}^{dxz}$, **P** matrix $\in \mathbf{R}^{axt}$

4A3. Results

The above algorithm was evaluated on two of the datasets above, namely **AwA** and **aPaY**, as for **Caltech-UCSD Birds** dataset, word vectors for many bird classes were absent.

Deep Features (p@k)	AwA Dataset (% acc.)	aPaY dataset (% acc.)
GoogLeNet (p@1)	26.89	20.53
GoogLeNet (p@3)	41.32	38.88
GoogLeNet (p@5)	64.88	58.09

4A4. Possible Shortcomings

Although the kNN search step while finding test classes performed correctly for p@5 results, obtaining an accuracy of 64.88 and 58.09 in AwA and aPaY respectively clearly defies any randomness in the result, it was not able to beat the benchmark results obtained in [cite some papers]. One of possible reasons could be that linear projection fails to adequately capture the relationship between images and attributes and that between attributes and wordvectors. The other possible reason could also be that no significant relations exist between attributes and wordvectors, but there could be an association between images and attributes and that between images and wordvectors as depicted in previous works. The small size of attribute matrix could also be a possible explanation of insignificant results, as they would not have been able to provide much of discriminative information.

4A5. Technology

Python libraries: Scipy, Numpy, SK-Learn etc.
Environment : Xeon Processors, 24 cores CPU

4B. Formulating a nonlinear model with Artificial Neural Nets

4B1. Images to Word-Vectors projection with the intermediate projection being Attributes

4B1.1 Motivation

Failure of above experiment paved the path to look for a non linear relationship between images and attributes and then between attributes and wordvectors.

4B1.2 Algorithm

The visual features are nonlinearly projected to attributes, and then attributes are nonlinearly projected to wordvectors. Following loss function is minimized with different optimization algorithms for e.g. Adam [cite paper], Nesterov [Cite paper], Stochastic Gradient Algorithm (SGD), etc. available in python libraries.

The nonlinear network for visual features to attributes have **4** hidden layers with the number of neurons from first hidden layer to output layer being **2048**, **512**, **256**, **128**, ***a*** where ***a*** is dimension of the attribute vector per image. The input layer has number of neurons equal to ***d*** where ***d*** is the dimension of visual features.

The second complementary network has **2** hidden layers with the number of neurons from first hidden layer being **128**, **256** and the output layer has ***t*** neurons where ***t*** is the dimension of the word-vectors. The input layer has ***a*** number of neurons where ***a*** is the dimension of the attribute vector.

Input : Deep Visual Features, Attributes of images (note that this time only training class images attributes are fed) and training class word-vectors.
Output : A Nonlinear Neural Network model to predict attributes from images and another Nonlinear Neural Network model to predict word-vectors from the predicted attributes.

4B1.3 Result

Deep Features (p@k)	AwA (% acc.)	aPaY (% acc.)
CNNM2K (p@1)	27.19	21.11
CNNM2K (p@3)	42.65	39.87
CNNM2K (p@5)	65.88	60.12
GoogLeNet (p@1)	28.12	22.18
GoogLeNet (p@3)	44.87	42.87
GoogLeNet (p@5)	67.11	61.12

4B1.4 Possible Shortcomings

Poor results as seen in Table [] hint that including both semantic space information, attributes and wordvectors does no good to the model. Although increasing the number of possible choices for words during kNN search step (after the test wordvectors have been predicted) does increase accuracy, but these results should not be considered significant as it is due to increased number of options. (p@1 is specifically very low). More specifically learning a linear or nonlinear projection from images to attribute space and then from attribute space to wordvector space fails to obtain significant results. So this idea should be abandoned. One of the reasons for this failure could be insufficient training data for learning a nonlinear model to learn projections from attribute to wordvectors. As the number of classes are limited and less than 100, the attribute vectors and wordvectors for those classes are also limited and thus running a neural network optimization algorithm for less training data multiple number of times may have led to overfitting, thus not able to generalize well to test classes images.

4B1.5 Technology

Python Libraries : Numpy, Scipy, SK-Learn, Keras, Tensorflow
Environment : Xeon Processors, 24 cores CPU

4B2: Formulating a nonlinear model to predict attributes individually from visual features

4B2.1: Motivation

Failure of above experiments to club both the semantic spaces together motivated for solely focus on attributes. One can argue that nonlinear projections from images to wordvectors could also have been learned, but this approach was already explored by the authors in [cite paper]. Although the one presented in this experiment was not looked into by authors active in this area. Moreover, this nonlinear system of learning attributes from images is closer to the way a human brain perceives and processes information.

4B2.2: Algorithm

A nonlinear neural network with the above setting was used to learn projections from visual features to attributes. Two variations were taken into consideration. One being, the projection of entire ***d*** dimensional deep visual features to complete ***a*** dimensional attribute vector, and other being the projection of ***d*** dimensional visual features to a single attribute out of ***a***ttributes. This for latter setting ***a*** Neural Networks was trained for ***a*** different attributes. Interestingly both the settings result in accuracies which differ by significant value, where the latter model outperforms the former one.

Input: Deep Visual Features, Attribute Vector of each image
Output: A nonlinear model to predict attributes from images.

4B2.3 Result

Deep Features	AwA (% acc.)	aPaY (% acc.)	CUB (% acc.)
CNNM2K	56.17	53.22	54.91
GoogLeNet	58.25	54.24	56.11

4B2.4: Possible Shortcomings

This model outperformed the several existing approaches including the state-of-the-art model by Lampert et al. But it was not able to beat the newer models which incorporated Domain Adaptation [cite papers], Graph-based model construction [cite papers], etc. One possible explanation for the latter model where individual attributes are learned instead of learning an attribute vector as a whole is that it is easier to catch the relationship between images and individual attributes rather than capturing multiple information form images at once, i.e. a specific learning is always robust than a generalized one. As explained earlier, this model only learns projections from images to attribute space, with no extra intelligent processing done due to which it is unable to beat the newer models by other authors.

4B2.5: Technology

Python Libraries : Numpy, Scipy, SK-Learn, Keras, Tensorflow
Environment : Xeon Processors, 24 cores CPU

4C. Formulating an Evolutionary Neural Network by Neural Evolution of Augmenting Topologies algorithm

4C1. Motivation

Neural Evolution of Augmenting Topologies (**NEAT**) is an evolutionary genetic algorithm to evolve Neural Nets which adapt to the problem setting instead of being hand-crafted (which has a fixed framework). This characteristic of problem adaptation through genetic evolution makes NEAT suitable for developing complex nets which exploit the hidden information present in data. Learning the hidden relationship between attributes and images calls for the use of NEAT, and moreover, the way our visual system have evolved to recognize high-level characteristics in objects also attracts the biologically inspired NEAT to recognize high-level attributes in images.

4C2. Algorithm

The NEAT algorithm is implemented in Python language and like any other genetic algorithm needs a fitness function to be defined which is accordingly optimized. It also requires an initial population and other parameters which are mentioned in the configuration file the python-neat library provides. The fitness function is defined as the square loss function where the square loss of the difference of predicted attribute vector and actual attribute vector is minimized.

4C3. Result

Unfortunately, no result was obtained for any of the datasets since the iterations were very slow to converge in feasible time despite being parallelized on 24 core CPU.

4C4. Possible Shortcoming

As evident from the ***4C3. Result*** subsection, failure of obtaining any result is credited to very slow convergence rate. Genetic Algorithms are infamous for converging slowly, and this particular instance of NEAT proved no exception to it. Since this slow convergence is inherent in the optimization of genetic algorithms, overcoming this shortcoming is outside the scope this research on subspace learning.

4C5. Technology

Python Libraries : Numpy, Scipy, SK-Learn, python-neat
Environment : Xeon Processors, 24 cores CPU

4D. Formulating a deep CNN model to learn attributes directly from images

4D1. Motivation

We humans visualize the object and detect characteristics of it through our visual perception system. **Convolutional Neural Networks** (**CNNs**) are the most accurate and advanced learning systems which mimic the human perception system. Raw pixel images are fed into it, rather than the visual features, such that it extracts the most suitable visual features for the current job in hand (which is learning attributes). Since CNN mimic our eyes, it is but obvious to use it to identify high-level characteristics in images.

How a CNN works, lies in its ability to identify low-level characteristics in images likes edges, curves, textures, colors and later building complex high-level structures like a circle, sphere, etc. from these low-level features in higher layers, which is approximately how our visual cortex in brain works. This ability motivated the use of CNN to identify attributes in images.

4D2. Algorithm

For benchmarking purposes, **AlexNet** [cite paper] was implemented. The differently sized images were resized to 200 x 200 pixels size. The arrangement of layers is done similar to AlexNet, only the size of convolutional layers, pooling layers and stride vary, thus the structure of CNN remains same.

The structure followed is explained below:

1> Input layer of size 200 x 200 pixels
2> Convolutional Layer (96 filters of size 5 x 5)
3> Activation layer (RELU Activation function)
4> Convolutional Layer (256 filters of size 5 x 5)
5> Activation Layer (RELU Activation function)
6> Convolutional Layer (384 filters of size 3 x 3)
7> Activation Layer (RELU Activation function)
8> Max Pooling (pool size = 2 x 2, with stride = 2,2)
9> Convolutional Layer (256 filters of size 3 x 3)
11> Activation Layer (RELU Activation function)
12> Fully Connected Layer (size 4096)
13> Activation Layer (RELU Activation function)
14> Fully Connected Layer (size 1024)
15> Activation Layer (RELU Activation function)
16> Fully Connected Layer (size ***a,*** where ***a*** is the size of attribute vector)
17> Activation Layer (Softmax Activation function)

There's a slight difference from AlexNet in terms of less Max-Pooling layers and usage of more RELU Activation function. This is done to decrease the loss of information due to Pooling (at the expense of increased computation time) and introducing more nonlinearity in the model respectively. One may argue that the last Fully Connected layer nonetheless extracts deep features from images (and this is how the deep visual features used in previous experiments were extracted from popular CNN nets) and is thus used for learning the relationship between images and attributes, a task which is similar to experiment 3, but one should remember that the deep visual features used in those experiments were extracted from nets whose primary criterion was classification of images in one of the ***n*** classes. Here, however, we do not classify the images into classes, but learn to identify attributes, thus the low-level features and subsequently high-level features produced in deeper layers of the network are completely different from those produced in popular nets, whose last fully connected layer provides deep visual features.

4D3. Result

Due to lack of enough computation power and proper GPU setup, no result for any dataset was obtained.

4D4. Possible Shortcomings

Each single epoch or iteration over the complete dataset of aPaY takes more than an hour on 140 cored CPU. This leaves very little chance to tune the parameters of the network, the optimization algorithms and the hyper parameters of it. There was a lack of suitable computation environment, to explore for possible shortcomings in this model.

4D5. Technology

Python Libraries : Numpy, Scipy, SK-Learn, Keras, Tensorflow
Environment : Xeon Processors, 24 cores CPU

E. CONCLUSION

This thesis report presents two different settings for subspace learning, for which various algorithms were presented. The two settings are subspace learning in classical image classification setting and intermediate semantic subspace learning in Zero-Shot Learning (ZSL) framework. Research done to improve the subspace learning via intelligent fusion of multiple modals shows efficacy in the improved results over **NUS Object** and **NUS Scene** datasets. The process of early fusion was followed along with incorporating label information in the fusion process, thus obtaining an informed latent subspace which has richer discriminative information embedded in it.

In the short survey presented here in this thesis, a brief overview of ZSL is presented where an intuitive definition to the problem statement is furnished and important research directions in it are covered. Several approaches to it are classified into two broad categories namely Attribute based information transfer and Word Vector-based data driven representations. New algorithms proposed in this thesis for ZSL framework were gradually developed from naive approaches to more complex approaches ending with efforts to develop a CNN based model to tackle attribute learning directly from images. Few of the experiments did beat the results obtained in existing papers and few failed to be experimented well due to lack of sufficient computation power.

A discussion of the shortcomings in the existing approaches as well as in the novel ones is also presented, and few possible research directions are also pointed out. A continually increasing number of works suggest that this field still yearns for major contributions from machine learning community.

F. REFERENCES

[1] S. Akaho. A kernel method for canonical correlation analysis. CoRR, abs/cs/0609071, 2006. [2] P. K. Atrey, M. A. Hossain, A. El Saddik, and M. S. Kankanhalli. Multimodal fusion for multimedia analysis: a survey. Multimedia Systems, 16(6):"345–379", 2010.
[3] P. N. Belhumeur, J. a. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. IEEE Trans. Pattern Anal. Mach. Intell., 19(7):711–720, July 1997.
[4] J. C. Caicedo, J. BenAbdallah, F. A. Gonza´lez, and O. Nasraoui. Multimodal representation, indexing, automated annotation and retrieval of image collections via non-negative matrix factorization. Neurocomput., 76(1):50–60, Jan. 2012.
[5] J. C. Caicedo and F. A. Gonza´lez. Multimodal fusion for image retrieval using matrix factorization. ICMR '12, pages 56:1–56:8. ACM, 2012.
[6] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng. Nus-wide: A real-world web image database from national university of singapore. In Proc. of ACM Conf. on Image and Video Retrieval (CIVR'09), Santorini, Greece., July 8-10, 2009.
[7] R. Datta, J. Li, and J. Z. Wang. Content-based image retrieval: Approaches and trends of the new age. In Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval, MIR '05, pages 253–262. ACM, 2005.
[8] A. Eweiwi, M. S. Cheema, and C. Bauckhage. Discriminative joint non-negative matrix factorization for human action classification. In Pattern Recognition - 35th German Conference, GCPR, pages 61–70, 2013.
[9] R. Gaujoux and C. Seoighe". Semi-supervised nonnegative matrix factorization for gene expression deconvolution: A case study. Infection, Genetics and Evolution, 12(5):913 – 921, 2012. [10] R. Gaurav, A. Vallecha, M. Verma, and K. K. Shukla. Multimodal subspace learning on flickr images. In 2015 IEEE UP Section Conference on Electrical Computer and Electronics (UPCON), Dec 2015.
[11] Q. Gu, Z. Li, and J. Han. Joint feature selection and subspace learning. IJCAI'11, pages 1294–1299. AAAI Press, 2011.
[12] S. K. Gupta, D. Phung, B. Adams, T. Tran, and S. Venkatesh. Nonnegative shared subspace learning and its application to social media retrieval. 16th ACM SIGKDD KDD '10, pages 1169–1178, 2010.
[13] S. K. Gupta, D. Phung, B. Adams, and S. Venkatesh. Regularized nonnegative shared subspace learning. Data Min. Knowl. Discov., 26(1):57–97, Jan. 2013.
[14] X. He. Locality Preserving Projections. PhD thesis, Chicago, IL, USA, 2005. AAI3195015. [15] X. He, D. Cai, S. Yan, and H.-J. Zhang. Neighborhood preserving embedding. In Tenth IEEE, ICCV '05, pages 1208–1213, Washington, DC, USA, 2005. IEEE Computer Society.
[16] J. R. Kettenring. Canonical analysis of several sets of variables. Biometrika, 58(3):433–451, 1971.
[17] D. Kitamura, H. Saruwatari, Y. Iwao, K. Shikano, K. Kondo, and Y. Takahashi. Superresolution-based stereo signal separation via supervised nonnegative matrix factorization. In Digital Signal Processing (DSP), 2013 18th International Conference, pages 1–6, July 2013.
[18] L. Lahti and O.-P. Huovilainen. dmt: Dependency Modeling Toolkit, 2013. R package version 0.8.20.

[19] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In In NIPS, pages 556–562. MIT Press, 2000.
[20] H. Liu, R. Ji, Y. Wu, and G. Hua. Supervised matrix factorization for cross-modality hashing. CoRR, abs/1603.05572, 2016.
[21] H. Liu, Z. Wu, X. Li, D. Cai, and T. Huang. Constrained nonnegative matrix factorization for image representation. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 34(7):1299–1311, July 2012.
[22] J. Liu, C. Wang, J. Gao, and J. Han. Multi-View Clustering via Joint Nonnegative Matrix Factorization. In Proc. of 2013 SIAM Data Mining Conf. (SDM'13), 2013.
[23] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: a power tool for interactive content-based image retrieval. IEEE Trans. Circuits Syst. Video Techn., (5):644–655, 1998.
[24] A. Sharma and D. W. Jacobs. Bypassing synthesis: Pls for face recognition with pose, low-resolution and sketch. In 2011 IEEE, CVPR '11, pages 593–600, Washington, DC, USA, 2011. IEEE Computer Society.
[25] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. Neural Comput., 12(6):1247–1283, June 2000.
[26] A. Tenenhaus and M. Tenenhaus. Regularized generalized canonical correlation analysis. Psychometrika, 76(2):"257–284", 2011.
[27] A. Tripathi, A. Klami, and S. Kaski. Simple integrative preprocessing preserves what is shared in data sources. BMC Bioinformatics, 9(1):"1–13", 2008.
[28] S. A. Vavasis. On the complexity of nonnegative matrix factorization. SIAM J. on Optimization, 20(3):1364–1377, Oct. 2009.
[29] J. Via, I. Santamaria, and J. Perez". A learning algorithm for adaptive canonical correlation analysis of several data sets. Neural Networks, 20(1):139 – 152, 2007.
[30] J. J.-Y. Wang and X. Gao. Max min distance nonnegative matrix factorization. Neural Networks, 61:75 – 84, 2015.
[31] K. Wang, W. Wang, R. He, L. Wang, and T. Tan. Multi-modal subspace learning with joint graph regularization for cross-modal retrieval. In 2nd IAPR ACPR 2013, Naha, Japan, November 5-8, 2013, pages 236–240, 2013.
[32] Q. Wang, H. Lv, J. Yue, and E. Mitchell. Supervised multiview learning based on simultaneous learning of multiview intact and single view classifier. CoRR, abs/1601.02098, 2016. [33] L. Z. Zhenfeng Zhu, Linlin Du and Y. Zhao. Shared subspace learning for latent representation of multi-view data. Journal of Information Hiding and Multimedia Signal Processing, Vol. 5, No. 3, pp. 546-554, July 2014.
[34] N. Zhou, Y. Xu, H. Cheng, J. Fang, and W. Pedrycz. Global and local structure preserving sparse subspace learning: An iterative approach to unsupervised feature selection. Pattern Recognition, pages –, 2015.
[35] Z. Zhou and F. Schwenker, editors. Partially Supervised Learning - Second IAPR International Workshop,PSL, volume 8183. Springer, 2013.