A Data-Driven Study of Image Feature Extraction and Fusion

Zhiyu Wang[a], Peng Cui[a], Fangtao Li[b], Edward Chang[b], Shiqiang Yang[a]

[a] *Tsinghua National Laboratory for Information Science and Technology*
Department of Computer Science, Tsinghua University. Beijing, China
Beijing Key Laboratory of Networked Multimedia, Tsinghua University, China

[b] *Google Beijing*

Abstract

Feature analysis is the extraction and comparison of signals from multimedia data, which can subsequently be semantically analyzed. Feature analysis is the foundation of many multimedia computing tasks such as object recognition, image annotation, and multimedia information retrieval. In recent decades, considerable work has been devoted to the research of feature analysis. In this work, we use large-scale datasets to conduct a comparative study of four state-of-the-art, representative feature extraction algorithms: color-texture codebook (CT), SIFT codebook, HMAX, and convolutional networks (ConvNet). Our comparative evaluation demonstrates that different feature extraction algorithms enjoy their own advantages, and excel in different image categories. We provide key observations to explain where these algorithms excel and why. Based on these observations, we recommend feature extraction principles and identify several pitfalls for researchers and practitioners to avoid. Furthermore, we determine that in a large training dataset with more than $10,000$ instances per image category, the four evaluated algorithms can converge to the same high level of category-prediction accuracy. This result supports the effectiveness of the data-driven approach. Finally, based on learned clues from each algorithm's confusion matrix, we devise a fusion algorithm to harvest synergies between these four algorithms and further improve class-prediction accuracy.

Keywords: image annotation; image feature; web-scale; fusion

1. Introduction

Extracting useful features from a scene is an essential subroutine in many multimedia data analysis tasks such as classification and retrieval. Remarkable progress has been made in multimedia computing, computer vision and signal processing in recent decades. Despite this finding, it is still notably difficult for computers to accurately recognize an object or analyze the semantics of a

scene. For example, suppose that we want to recognize a piece of white paper in an image. A naive feature we can use is "a white two dimensional rectangle." However, such a feature will not work in most cases because of the following:

1. The paper may be folded.
2. The viewing angle of the piece of paper may not perpendicular, and hence the paper does not appear to be rectangular.
3. Environmental factors such as occlusion and lighting can cause changes in its shape and color.

The above challenges are all related to *feature invariance* issues. A second challenge is called *feature aliasing* or *feature selectivity*: how well a feature can differentiate one object from the others. For example, the feature "white two-dimensional rectangle" can be used to describe many other objects: a piece of white cloth, a white table, and a white wall, among others. The goal of feature extraction is to find features that are both *invariant* and *selective*.

All traditional feature extraction approaches focus on some specific information in the image. For example, the color-texture codebook (CT) focuses on the statistics of colors and textures in small regions of an image. SIFT focuses on local invariant shapes. Recently, neuro-based approaches such as HMAX and convolution networks (ConvNet) have been proposed to model features according to how the human visual system extracts features. HMAX [45] builds computing models that use the pioneering neuroscience work of Hubel [22]. Hubel's work indicates that visual information is transmitted from the primary visual cortex (V1) through extrastriate visual areas (V2 and V4) to the inferotemporal cortex (IT). The IT, in turn, is a major source of input to the prefrontal cortex (PFC), which is involved in linking perception to memory and action [33]. The pathway from the V1 to the IT (called the *visual frontend*) consists of a number of simple (lower) and complex (higher) layers. The lower layers attain simple features that are invariant to scale, position and orientation at the pixel level. Higher layers can combine simple features to recognize more complex features at the object-part level. Pattern recognition at the lower layers is unsupervised, whereas recognition at the higher layers involves supervised learning. This particular neuroscience-motivated model appears to enjoy at least a couple of advantages: (1) it balances feature selectivity (at lower layers) and invariance (at higher layers), and (2) it models edges of an object and then combines edges to recognize parts of an object and place these features in a hierarchical context. Similar to HMAX, ConvNet is also a neuro-based approach. It differs from HMAX primarily in the way that ConvNet iterates more over the data to learn a model with a deep architecture [39]. This allows for the capture of both the structure and detail of an object.

Herein, we perform comparative evaluation that demonstrates that different feature extraction algorithms have their own set of advantages and excel in different image categories. We provide key observations about why certain algorithms perform better with different image categories. Based on these observations, we

establish feature extraction principles and identify several pitfalls for researchers and practitioners to avoid:

1. When training data are insufficient, no scheme performs well. However, because simple algorithms such as CT and SIFT do not require much data to learn model parameters, they may be a better choice when training data are scarce.
2. Increases in the amount of training data correlate with a jump in the accuracy of complex models, such as HMAX and ConvNet. Different feature extraction algorithms enjoy their own advantages, and excel in different image categories.
3. When training data are abundant, all the four algorithms, simple or complex, converge to the same level of accuracy.

The major contributions of this paper are summarized as follows:

1. Through our comparative analysis, we identify pitfalls of past studies: either they did not use enough training data, or their testbed composition already favors a particular feature extraction algorithm.
2. Through our large-scale comparative study, we demonstrate the benefit of employing large training datasets in training, which can make both simple and complex algorithms converge to the same level of accuracy.
3. We devise a fusion algorithm based on learned clues from each algorithm's confusion matrix. Our algorithm harvests synergies between these four algorithms and further improves class-prediction accuracy.
4. We established a large testbed for the research community, namely an annotated dataset of six million PicasaWeb images, which will be released publicly with this paper.[1]

The rest of the paper is organized as follows. Section 2 surveys the related work. Section 3 briefly introduces the four feature extraction algorithms evaluated in this paper. Section 4 details an algorithm that fuses multiple feature extraction methods that we demonstrate can perform better than any individual feature extraction scheme alone. Section 5 explains the setup of our experiments and presents their results. Finally, we offer concluding remarks in Section 6.

2. Related Work

The multimedia community has been striving to bridge the *semantic gap* [20, 46, 62] between low-level features and high-level semantics for decades. (Comprehensive surveys are given in [5, 20].) With high-quality image features, fancy applications can significantly improve a user's experience [9, 11, 30]. One key

[1] Downloads at https://sites.google.com/site/picasawebdataset/home

problem is how to extract powerful features. Numerous feature extraction algorithms have been developed for image annotation [46], as well as machine learning algorithms [14, 36, 59, 60, 61]. Roughly speaking, image features can be grouped into four types: color, texture, local features, and shapes. Color features are the most straightforward image feature and therefore were the first to receive sufficient study. Many color-based image retrieval algorithms have been developed [18]. Typical color features include the color histogram [19], color invariance [17], and color saliency [51]. Texture features, such as local binary patterns (LBP) [34], pyramidal-structured wavelet transforms [37], and tree-structured wavelet transform [8], are another significant set of signals for recognizing objects. Color and texture features are usually combined to solve image retrieval problems. Color-texture histograms are widely used for object tracking and recognition. After a decade of using color and texture as the main features for an image, a breakthrough came with the scale-invariant feature transform (SIFT) [32]. SIFT was a large step forward for extracting scale-invariant features. SIFT features are local features, and have been demonstrated to be effective in detecting near-replicas of images. SIFT has often been combined with unsupervised learning algorithms to solve multiple image processing problems. Recently, Y. Bengio [2] proved that it is theoretically impossible to represent some functions by architectures that are too shallow. In response to this finding, researchers learned from neuroscientists how the human visual system works [22], and it has been proven to be a deep architecture. Since Hinton's work in 2006 [21], deep models have been developed with good results. The earliest deep architectures include self-organizing neural networks [15] and the predecessor of the convolutional network (ConvNet), which is applied in document recognition [28]. For natural image processing, one of the representative works is HMAX [45], which strikes a good balance between feature selectivity and feature invariance. Furthermore, ConvNet [39] was introduced to establish a deep unsupervised learning architecture to learn powerful features. These deep learning features are based on edge detection and thus focus on the shapes of objects. Because our evaluation covers color, texture, local feature and shape, the conclusions that we draw hold for any subset of these combinations.

Though many comparative studies on features have been performed in the past, they generally suffer from a couple of limitations. First, most studies employ a relatively small training dataset; this results in better feature extraction algorithms not having the opportunity to demonstrate their superiority with training data increases. Second, some comparative studies' results are highly dependent on the makeup of the testbed. If categories of images favoring a particular algorithm dominate a testbed, that algorithm certainly achieves the best result. Unfortunately, most studies do not pay attention to these two limitations, which results in their conclusions having limited applicability. More importantly, we focus on studying representative state-of-the-art algorithms using large datasets. Our dataset size is much larger than that used in previous attempts (e.g., [13]) for data-driven image classification [42, 63].

Because different feature extraction algorithms have different advantages in dif-

ferent scenarios, it is a natural consideration to integrate different methods together to overcome any drawbacks from one individual method that may be an advantage to another [1, 56]. This fusion of methods leads to enhanced final performance. There are various ways to combine multiple algorithms. These traditional schemes aim to improve class-prediction accuracy by applying statistical learning to the same set of features. In our work, we pay attention to employing different features, i.e., different views provided by different feature-extraction algorithms. Our fusion method uses a confusion matrix to guide the choice of the best feature sets (views) into the raw images and then makes a collective class-prediction decision. Our approach differs from the traditional ensemble schemes in two respects. First, most traditional fusion schemes combine several feature sets into a single one and then use a statistical algorithm to learn the best feature combination [10]. Second, most ensemble schemes work on the same feature space and use statistical methods to mask prediction error. Our approach builds on these schemes and emphasizes that we take advantage of both multiple feature and statistical methods to reduce prediction error.

In general, ensemble methods can be classified into two groups: *feature-level* and *decision-level*. The first group pays focuses on combining features. When there are multiple feature sets, one of the most straightforward ways to combine features is to fuse on the feature-level. Algorithms of this type include stitching features together to create a super feature set [49] and constructing a classifier from the super feature set in conjunction with some learned cost [52]. Giridharan et al. proposed a discriminative approach to fusion of multimodal features of a video [23]. Wang et al. proposed a smart approach to fusion of multiple visual features in a graph-based learning scheme [53, 54]. However, these methods may encounter the curse of dimensionality or be confined by a feature-dependent structure [57], which is hard to generalize to different feature sets. The second group, decision-level fusion [38], is a good method to avoid the curse of dimensionality. Decision-level fusion can be implemented by either voting or constructing features with meta-classifier results. For instance, pairwise coupling, tree-structure ensemble, and error-correction output coding [41] all attempt to work on the same feature set (the same feature space) with robust statistical methods to minimize class-prediction error. Feature-level fusion and decision-level fusion have also been compared using video datasets, and the performance is related to semantic concepts [48]. In contrast to the feature-level and decision-level approaches, our fusion method conducts statistical inference on multiple views (through different features generated by different algorithms) of the raw images. As each view enjoys its own pros and cons, we employ confusion matrices to guide the class-prediction process through the least inter-class-confusion inference path. This approach not only avoids the curse of dimensionality as in feature-level ensemble application but also displays a superior before to decision-level methods by using advantages from different feature extraction algorithms.

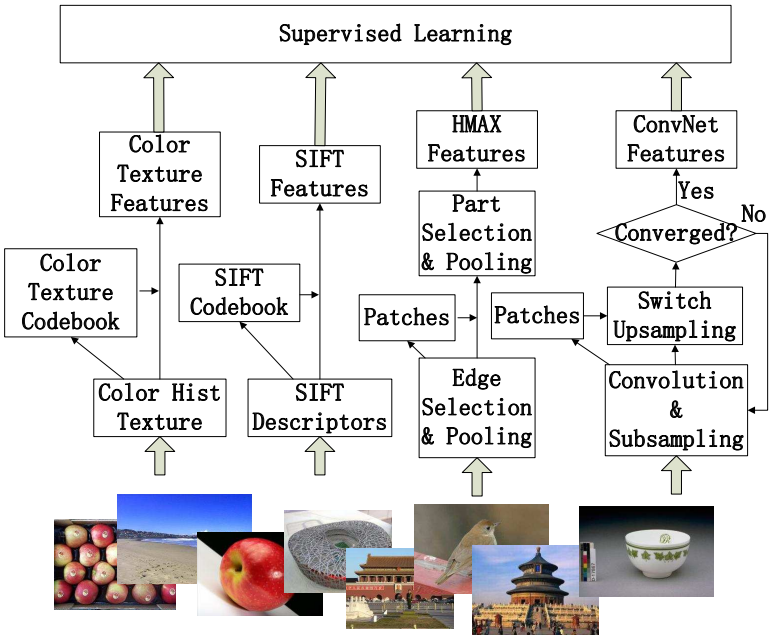

Figure 1: A framework of image classification including four pathways.

3. Feature Extraction Algorithms

In this section, we present four representative algorithms for image feature extraction: *color-texture codebook (CT)*, *SIFT codebook*, *HMAX*, and *Convolution Networks (ConvNet)*. Figure 1 depicts a framework that consists of these four algorithms. The input to the framework is a set of images. After extracting descriptors such as color-texture histograms, SIFT descriptors, HMAX edges, and ConvNet's encoding results, the framework conducts an unsupervised learning stage to learn codebooks or patch pools. Then, by matching the low-level descriptors to the codebooks/patches, the final output of each algorithm can be directly applied as the input to a supervised learning machine. This supervised learning machine performs class prediction for each input image. To facilitate further analysis of these four feature extraction algorithms, we outline color-texture codebook in Section 3.1, SIFT codebook in Section 3.2, HMAX in section 3.3, and ConvNet in Section 3.4.

Descriptions of the variables throughout this work are given in Table 1. We use bold characters for matrices and vectors and plain characters for values.

3.1. Color-Texture Codebook

In the color-texture (CT) codebook algorithm, images are characterized by three perceptual features: color, texture, and shape. These perceptual features are

Table 1: Variable definitions.

Variable	Explanation
$\mathbf{I}$	the image data
$Gauss(x, y, \sigma)$	Gaussian value at some point
$L(x, y, \sigma)$	Gaussian blurred value at some point
$D(x, y, \sigma)$	difference of Gaussian at some point
D_{xx}, D_{xy}, D_{yy}	partial diff of $\mathbf{D}$
$m(x, y)$	gradient magnitude
$\theta(x, y)$	orientation
t_{th}, r_{th}	thresholds for parameters
r_{th}	thresholds for parameters
$\mathbf{I}_{S\_edge}$	edge selection result
$\mathbf{I}_{edge}$	edge extraction result
$\mathbf{P}$	a patch
$\mathbf{I}_{S\_part}$	part selection result
$\mathbf{v}_{part}$	part extraction result
$\mathbf{F}_C$	filter for convolution layer
$\mathbf{C}$	output of a convolution layer
h	Sigmoid function
$\mathbf{v}_{convnet}$	ConvNet feature
$\mathbf{A}$	confusion matrix
A_{ij}	value of confusion matrix at i,j
$\mathbf{X}^{tr}$	features for training data
$\mathbf{Y}^{tr}$	labels for training data
$\mathbf{X}^{v}$	features for validating data
$\mathbf{Y}^{v}$	labels for validating data
$\mathbf{X}^{te}$	features for testing data
$\mathbf{Y}^{te}$	labels for testing data
G_i^f	group for feature f, category i

considered to be low-level descriptors of small image regions (also called blocks). The higher-layer feature-to-object mappings require a learning or clustering process. There have been many proposed methods to represent color, texture, and shape. What we describe in this subsection is a typical method that has been demonstrated to be competitive in this representation family [50].

3.1.1. Color Feature Extraction

Although the wavelength of visible light ranges from 400 nanometers to 700 nanometers, research effort shows that the colors that can be named by all cultures are generally limited to eleven [6]. In addition to *black* and *white*, the discernible colors are *red*, *yellow*, *green*, *blue*, *brown*, *purple*, *pink*, *orange*, and *gray*. Thus, the color-texture codebook algorithm first divides colors into 12 color bins: 11 bins for the above colors and one bin for outliers. At the

coarsest resolution, we characterized color using a color mask of 12 bits. To record color information at finer resolutions, we record eight additional features for each color. These eight features are color histograms, color means (in H, S, and V channels), color variances (in H, S, and V channels), and two shape characteristics: elongation and spread. Color elongation characterizes the shape of a color, and spread characterizes how that color scatters within the image [29]. We categorize color features in coarse, medium and fine resolutions.

3.1.2. Texture Feature Extraction

Texture is an important cue for image analysis. Studies have demonstrated that characterizing texture features in terms of structure, orientation, and scale (coarseness) fits well with models of human perception [35]. A wide variety of texture analysis methods has been proposed in the past. We chose a discrete wavelet transformation (DWT) using quadrature mirror filters [47] because of its computational efficiency.

Each wavelet decomposition of a 2D image yields four subimages: a $\frac{1}{2} \times \frac{1}{2}$ scaled-down image of the input image and its wavelets in three orientations: horizontal, vertical and diagonal. Decomposing the scaled-down image further, we obtain a tree-structured or wavelet packet decomposition. The wavelet image decomposition provides a representation that is easy to interpret. Every subimage contains information on a specific scale and orientation and also retains spatial information. We obtain nine texture combinations from subimages of three scales and three orientations. Because each subimage retains the spatial information of texture, the CT algorithm also computes the elongation and spread of each texture channel. For further details, please consult [25, 50].

3.1.3. Color-Texture Codebook Construction

Many researchers suggest the addition of an unsupervised layer between raw features and semantics. The *bag of words* (also known as *bag of features*) is one widely used model and is suitable for clustering color and texture histograms into a codebook [25] to reduce feature dimension.

3.2. SIFT Codebook

A SIFT [31] feature is a typical designed feature that is robust against orientation, scale and location variance of an image. To apply SIFT features to image classification, one needs first to extract SIFT descriptors and then cluster those descriptors into a codebook [4, 24]. With the codebook, a feature of fixed length can be extracted that meets the input requirement of a supervised learning algorithm.

3.2.1. SIFT Descriptor Extraction

Four principle stages are conducted to extract invariant SIFT descriptors.

Scale-space Extrema Detection. A Difference-of-Gaussian (DoG) function is used to efficiently search the image on all scales and locations for interesting points or keypoints. Keypoints are identified as local minima/maxima of the DoG images across scales. A DoG image at point (x, y) is given by

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{1}$$

where $L(x, y, k\sigma)$ is calculated by the convolution of the original image and a Gaussion blur at scale $k\sigma$, i.e.,

$$L(x, y, k\sigma) = Gauss(x, y, k\sigma) * I(x, y) \tag{2}$$

and

$$Gauss(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}, \tag{3}$$

where k is a constant multiplicative factor.

Keypoint Localization. For each candidate location of key points, location and scale information are determined using a detailed model. Only stable key points are kept. The strategies that determine stability include the interpolation of nearby locations for accurate position, the elimination of low-contrast key points, and the removal of edge responses. The quadratic Taylor expansion of the Difference-of-Gaussian scale-space function is given by

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}, \tag{4}$$

where $\mathbf{x} = (x, y, \sigma)$. If the offset between the local maxima and this point is too large, it is assumed that this local maxima is closer to another key point. Otherwise, if the second-order of the Taylor expansion is less than a given threshold, the point is discarded.

If we analyze the second-order Hessian matrix

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{5}$$

we can calculate $Tr(\mathbf{H}) = D_{xx}+D_{yy} = \alpha+\beta$ and $Det(\mathbf{H}) = D_{xx}D_{yy}-(D_{xy})^2 = \alpha\beta$. By solving for α and β and ensuring $|\alpha| > |\beta|$, we can calculate $r = \alpha/\beta$. If $(r+1)^2/r > (r_{th}+1)^2/r_{th}$, the point is discarded because of edge effects.

Orientation Assignment. Each keypoint is assigned with one or more orientations based on local image gradient directions. The invariance of orientation, scale, and location is guaranteed because all operations are relative to these

transformations. The Gaussian-smoothed image $L(x, y, \sigma)$ is taken so that all computations are scale-invariant in nature. For an image sample $L(x, y)$ at scale σ, the gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$ are given by equations 6 and 7, respectively, where $L_{x,y}$ is the same as $L(x + 1, y)$.

$$m(x, y) = \sqrt{(L_{x+1,y} - L_{x-1,y})^2 + (L_{x,y+1} - L_{x,y-1})^2} \tag{6}$$

$$\theta(x, y) = tan^{-1}(\frac{L(x, y + 1) - L(x, y - 1)}{L(x + 1, y) - L(x - 1, y)}) \tag{7}$$

Keypoint Descriptor. Around each keypoint, local image gradients are measured. Shape distortion and change in illumination are invariant here. Orientation histograms are created in 4×4 pixel neighborhoods with each pixel having 8 bins, so the descriptor has a dimension of 128. The vector is normalized to unit length to enhance invariance to affine changes in illumination. Four principle stages are conducted to extract invariant SIFT descriptors. For further details, please consult [31, 32].

3.2.2. SIFT Codebook Construction

Because each image has various numbers of interesting points and thus have different stable SIFT features, we need to normalize the feature length when we classify images. A typical way to do this is to cluster descriptors into several representative centers, and then use those centers to measure the image instead of a direct comparison of SIFT descriptors.

3.2.3. SIFT Feature Extraction

When classifying images, one must first explore a large population of images and extract their SIFT descriptors. The SIFT descriptors are then clustered into a codebook, which is the state-of-the-art conversion algorithm for applying SIFT to obtain high level abstracted features [24]. After completing a codebook, features can be extracted from new images by matching their SIFT descriptors to corresponding entries in the codebook.

3.3. HMAX

The HMAX [45] algorithm is depicted in Figure 2. The figure shows that visual information is transmitted from the primary visual cortex (V1) through the extrastriate visual areas of the brain (V2 and V4) and then to the inferotemporal cortex (IT). Physiological evidence indicates that the cells in V1 largely conduct *selection* operations, and cells in V2 and V4 conduct *pooling* operations. Based on this, M. Riesenhuber and T. Poggio establish a feed-forward theory of object recognition [40] that provides a qualitative way to model the ventral stream

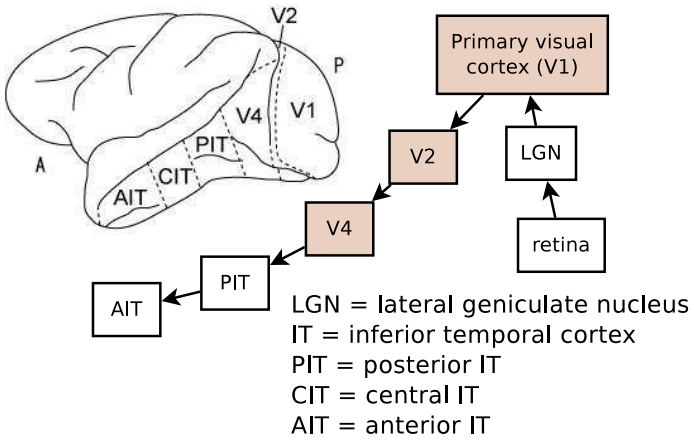

Figure 2: Information flow in the visual cortex. (See the brain structure in [58].)

in the visual cortex. Their model suggests that the visual pathway consists of multiple layers of computational units where *simple* S units alternate with *complex* C units. The S units address signal selectivity, whereas the C units address invariance. Lower layers recognize features that are invariant to scale, position, and orientation at the pixel level. Higher layers detect features at the object-part level from the combination of lower layer features. Pattern reading at the lower layers is largely unsupervised, in contrast to recognition at the higher layers, which involves supervised models.

Motivated by both physiological evidence [45] and computational learning theories, the HMAX pipeline consists of five steps:

- *Edge selection*. Edge selection corresponds to the operation conducted by cells in V1 and V2 [27], which detect edge signals at the pixel level.
- *Edge pooling*. Edge pooling also corresponds to cells in V1 and V2. The primary operation is to pool strong, representative edge signals using a *max* operator.
- *Unsupervised Learning*. To prevent too many features, which can lead to the dimensionality curse, or too shallow architecture, which may lead to trivial solutions, this unsupervised steps groups edges into patches to make the patches both representative of shapes and without duplication.
- *Part selection*. We model V2 to V4 step to look for image patches matching prototypes (patches) produced in the previous step.
- *Part pooling*. Cells in V4 [16] have larger receptive fields than V1 and address object-parts. Because of their large receptive fields, V4's selectivity is preserved over translation.

For further details about HMAX details, please consult Serre's thesis [44].

3.4. Convolutional Networks

Convolutional networks [28] combine local receptive fields, shared weights and spatial or temporal sub-sampling to ensure shift, scale and distortion invariance. Convolutional networks are widely used in multiple image processing and machine learning tasks [43]. The input image is alternatively processed with convolutional layers and sub-sampling layers to obtain features. Convolutional layers and sub-sampling layers are similar to the *edge selection* and *edge pooling* steps of HMAX. Indeed, the ConvNet algorithm was inspired by neuroscience. With deep neural learning of multiple iterations, the output converges to one pixel for each set of parameters. Each of these one-pixel values of the final layer can be stitched together to form the final feature vector, which can then be applied directly as the input of a supervised learning algorithm. ConvNet learns patches for the convolutional layer, which is similar to the function of patches in HMAX. However, while HMAX picks patches randomly from the training data to obtain a uniform distribution from all training data, ConvNet attempts to learn the most informative patches for higher efficiency, which is the major difference between the two algorithms.

3.4.1. Convolutional Layer

Several weight matrices are trained so that each matrix convolves with the image to extract various features. Each weight matrix is a 5×5 matrix that connects every 25 pixel block in the input image. This operation is a 2D convolution: $\mathbf{C} = \mathbf{I} * \mathbf{F}_C$.

3.4.2. Sub-sampling Layer

Each unit of the sub-sampling layer is connected to a 2×2 neighborhood in the corresponding feature map in the convolutional layer. These four inputs are added together, multiplied by a trainable coefficient, and then added to a trainable bias. The results of the convolutional layer are then passed through a sigmoidal function. Because the 2×2 receptive fields do not overlap, feature maps in the sub-sampling layer are one-fourth of the area of the feature maps in the corresponding convolutional layer.

3.4.3. Deep Learning and Learning from Data

Convolutional layers and sub-sampling layers alternate processing of the image with different weights. The convolutional layer will ignore boundary values and thus reduce the data scale by several pixels in both width and height. The sub-sampling layer will reduce the data to one fourth of the original size. Thus, if we continuously apply convolutional layer and sub-sampling layer processing with the data to construct a deep learning architecture, the scale of the data can reduce to a single value.

There are multiple weights provided to the convolutional layer, so the final output is a vector. A deep learning procedure can convert the image into a vector

with multiple weights. The weights in ConvNet can all be learned from multiple iterations of feature extraction. With an updating stage in each iteration, the weights can be learned from the data [3], and the feature becomes better and better. For further details about convolutional networks, please consult [28, 39]

4. Fusion

The four feature-extraction algorithms produce features with different advantages and drawbacks. In this section, we outline a fusion algorithm to harvest synergies between these four algorithms to further improve class-prediction accuracy. This fusion algorithm not only takes advantages of multiple image features but also avoids the curse of dimensionality. More precisely, with each feature set, a classifier can be constructed to perform class prediction. By constructing a *confusion matrix* (defined shortly) for a feature set, we can learn whether the classifier can provide a reliable class-prediction for an unlabeled instance. If the confusion matrix, which is trained by a validation data set, indicates that the predicted top class (say, yellow bus) was often misclassified into another class (say, yellow bulldozer), then the class prediction should not be fully trusted. Instead, we should look for assistance from any classifiers that are not confused by the two classes to perform disambiguation. In other words, our fusion method provides a result if one of the classifiers produces a highly confident prediction. If confidence is low, we can check the confusion matrix for the possible confusion set and make use of other classifiers to help further judge the prediction, so the results can be more reliable. The overall procedure is presented in Figure 4. The details are provided in the following paragraphs.

4.1. Data Preparing

The fusion algorithm first uses the training data to train individual SVM classifiers on each feature set produced by the four feature-extraction algorithms. Figure 4 presents this data-preparing stage from steps #1 to #3, whose input and output are defined as in Figure 3

4.2. Confusion Matrix

A confusion matrix is an $M \times M$ matrix showing the predicted and actual classifications [26], where M is the number of categories. The confusion matrix is denoted by $\mathbf{A}$, where A_{ij} stands for the probability that the actual label i is predicted as label j. By predicting the validating data, the confusion matrix can be constructed by counting the number of pairs of predicted labels and actual labels. We normalize the confusion matrix by the sum of each row of the matrix so that $\forall i, j, A_{ij} \in [0, 1]$. A simple example of a confusion matrix and its normalization is in Table 2. A confusion matrix is constructed by counting the number of pairs of predicted labels and actual labels as in the left table, e.g., the third value on the second row indicates that there are four instances predicted

Input: M // # of categories
D // the length of a feature vector
δ // threshold to cut the gap
$\mathbf{F}$ // feature set {CT, SIFT, HMAX, ConvNet}
N^{tr}, N^{v}, N^{te} // # of training/validating/testing instances
$\{(x^{tr,f}, y^{tr}) | x^{tr,f} \in \mathbf{X}^{tr,f}, y^{tr} \in \mathbf{Y}^{tr}, f \in \mathbf{F},$
$\mathbf{X}^{tr,f} \in \mathbf{R}^{N^{tr} \times D}, \mathbf{Y}^{tr} \in \{y_i | i \in [0, M-1]\}\}^{N^{tr}}$
// training data with label
$\{(x^{v,f}, y^{v}) | x^{v,f} \in \mathbf{X}^{v,f}, y^{v} \in \mathbf{Y}^{v}, f \in \mathbf{F},$
$\mathbf{X}^{v,f} \in \mathbf{R}^{N^{v} \times D}, \mathbf{Y}^{v} \in \{y_i | i \in [0, M-1]\}\}^{N^{v}})$
// validating data with label
$\{x^{te,f} | x^{te,f} \in \mathbf{X}^{te,f}, \mathbf{X}^{te,f} \in \mathbf{R}^{N^{te} \times D} | f \in \mathbf{F}\}$
// testing data without label
w_1, w_2, w_3 // weights for selecting top classifier
Output: $\hat{\mathbf{Y}}^{\mathbf{te}} \in \{y_i | i \in [0, M-1]\}^{N^{te}}$ // predicted labels for testing data

Figure 3: Input and Output of Fusion

Table 2: A simple example of confusion matrix.

Category	Predicted		
Actural	1	2	3
1	8	1	1
2	0	6	4
3	1	4	5

Category	Predicted		
Actural	1	2	3
1	0.8	0.1	0.1
2	0	0.6	0.4
3	0.1	0.4	0.5

as category 3, but actually, they are category 2. Computing the sum of each row and normalizing each value by the row sum, the normalized confusion matrix is in the right table. Figure 4 presents the confusion matrix construction from steps #4 to #16.

4.3. Confusion Groups

An analysis of the confusion matrix shows that each algorithm encounters different confused category sets. For instance, the CT algorithm cannot tell the difference between a *computer keyboard* and *calculator*. The SIFT algorithm confuses a *school bus* and a *bulldozer*. When a label i is predicted, the possibility of the actual label being j is marked in the confusion matrix A_{ji}. By sorting $\{A_{ji} | j\}$ in descending order, we obtain the top possible labels by cutting off the tail at a large gap location, i.e., disregard all values starting at the first j' where $A_{j'-1,i} - A_{j',i} < \delta$, where δ is the cutting off threshold. In addition, the selected label should have a probability larger than $1/M$, or else a random selection would outperform it. A group G_i^f is constructed by collecting these top possible labels. Refer again to Table 2 as an example. If the category 3 is predicted by this classifier, and $\delta = 0.15$, the sorting will provide the order $0.5, 0.4, 0.1$ in the column. Because $0.5 - 0.4 < 0.15$ but $0.4 - 0.1 > 0.15$ (or because $0.1 < 1/M$, $M = 3$), we keep 0.5 and 0.4 but discard 0.1 and the later

Initialize:

for all $f \in \mathbf{F}$ **do**
  $c_f \leftarrow SVM(\mathbf{X}^{tr,f}, \mathbf{Y}^{tr})$ // train SVM classifier c_f
end for

Compute confusion matrix:

for all $f \in \mathbf{F}$ **do**
  $\mathbf{A}^f \leftarrow \mathbf{0}$ // confusion matrix set to zero
  for all $x^{v,f} \in \mathbf{X}^{v,f}$ **do**
    $\hat{y}^{v,f} \leftarrow c_f(x^{v,f})$ // predict validating data with classifier c_f
    $A^f_{y^v,\hat{y}^{v,f}} \leftarrow A^f_{y^v,\hat{y}^{v,f}} + 1$
  end for
  for all $i \in [0, m-1]$ **do**
    $S \leftarrow \sum_j A^f_{ij}$ // compute sum of row
    for all $j \in [0, m-1]$ **do**
      $A^f_{ij} \leftarrow A^f_{ij}/S$ // normalize by each row
    end for
  end for
end for

Generate confusion groups:

for all $f \in \mathbf{F}$ **do**
  for all $i \in [0, M-1]$ **do**
    $\mathbf{V} \leftarrow$ARG(SORT($\{A^f_{ji}|j\}$))
      // sort in descending order, get the corresponding labels
    $G^f_i \leftarrow \{V_0\}$ // the first is always in the set
    for all $j \in [1, M-1]$ **do**
      if $A^f_{i,V_j} > 1/M$ **and** $A^f_{i,V_{j-1}} - A^f_{i,V_j} < \delta$ **then**
        $G^f_i \leftarrow G^f_i \cup \{V_j\}$
      else
        break
      end if
    end for
  end for
end for

Classify with confusion matrix guided ensemble:

for all $x^{te} \in \mathbf{X}^{te}$ **do**
  for all $x^{te,f} \in \mathbf{X}^{te,f}$ **do**
    $\hat{y}^{te,f} \leftarrow c_f(x^{te,f})$ // predict testing data with classifier c_f
  end for
  for all $f \in \mathbf{F}$ **do**
    $s^f_1 \leftarrow \sum G^f_{\hat{y}^{te,f}}$ // overall coverage
    $s^f_2 \leftarrow$MAX($G^f_{\hat{y}^{te,f}}$)$-$MAX2($G^f_{\hat{y}^{te,f}}$)
      // gap between the first two
    $s^f_3 \leftarrow$MAX($G^f_{\hat{y}^{te,f}}$) // confidence of the top candidate
    $s^f \leftarrow w_1 s^f_1 + w_2 s^f_2 + w_3 s^f_3$
  end for
  $\mathbf{U} \leftarrow$ARG(SORT(s_f))
    // sort in descending order, get the corresponding indexes
  $G \leftarrow G^{U_0}_{\hat{y}^{te,U_0}}$ // candidates generated by predicted label
  for all $f \in F, f \neq U_0$ **do**
    for all $i \in G$ **do**
      $v^f_i \leftarrow \sum_{j \in G} A^f_{ij}$ // voting values for f on i
    end for
    $v_i \leftarrow \sum_{f \in F, f \neq U_0} v^f_i$
  end for
  $\hat{y}^{te} \leftarrow$ARG(MAX(v_i))
end for

Figure 4: Flow chart for fusion by confusion matrix

ones (if any). In our algorithm, we record categories 2 and 3 in the candidate confusion group. That is to say, when this classifier predicts 3, it is more likely to be actual category 2 or 3, i.e., $G_3^f = \{2, 3\}$. By generating the confusion groups of candidates, when a predicted label is given, it is reasonable to say the actual label is most likely in this generated group. Figure 4 presents confusion groups construction from steps #17 to #29.

4.4. Confusion Matrix-Guided Ensemble

Using classifier prediction results, we can narrow down the category scope. For example, a predicted label *computer keyboard* from a classifier constructed by CT may in reality be a *computer keyboard*, *cellular telephone*, or *calculator*. This type of fuzzy accuracy is the main source of the prediction errors. However, when one classifier provides the result *computer keyboard* with high confidence, we can be quite sure that the actual category cannot be *school bus* or *watermelon*. Because the confusion categories are sparse, confusion categories of high confidence from one method can be a good input for another classifier to correctly prune results down to the correct result. The narrowed scope of categories are then adopted by the second, third, and fourth classifiers to vote for the most confident category, which is the prediction label of the fused method. It is important to decide the order of the classifiers for this hierarchical architecture. For the first classifier, we wish to narrow down the candidate scope and at the same time minimize the information loss, so we take the following metrics to supervise the sorting of the classifiers:

- *Large coverage*. The coverage of possible labels should be large, i.e., we prefer a higher probability that the actual label is in the group. The false negative rate should be as small as possible.
- *Large gap*. The gap between the top category and the second category should be large. This indicates that the prediction result is more stable. Labels kept in the group should have a large gap with those outside, which indicate that the group is constructed distinguishably. This second gap is guaranteed by a proper threshold δ.
- *High first confidence*. The confidence of the first category should be high, which indicates that the prediction result is more reliable. The confidence here is defined by $A_{ii}^f / \sum_j A_{ji}^f$.

An evaluation function consisting of the above metrics should provide a good selection of the first classifier. Here we apply a linear-weighted combination $s^f = w_1 s_1^f + w_2 s_2^f + w_3 s_3^f$, where s_1^f, s_2^f, s_3^f are the total coverage, the gap, and the first confidence respectively, and w_1, w_2, and w_3 are the corresponding weights. After the first classifier, the remaining classifiers are applied to decide which of the candidate categories have the highest confidence in a weighted voting manner. The ones with the maximum confidence is returned when they

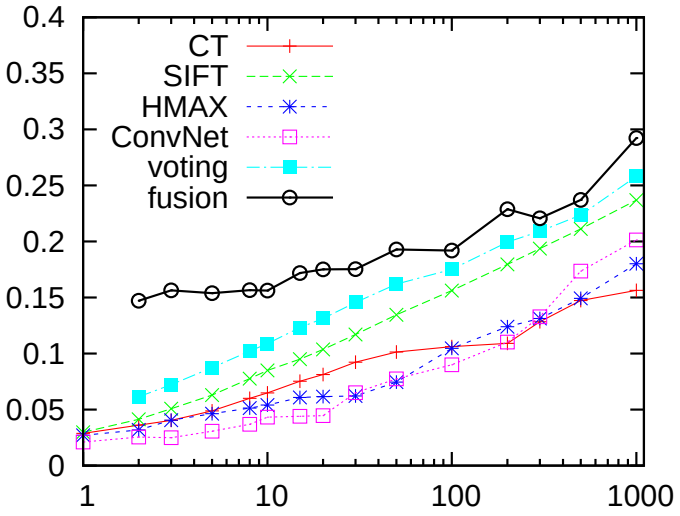

Figure 5: Accuracy for ImageNet dataset. X-axis shows the number of training instances for each category. Y-axis shows the average accuracy.

are the top instances for each category. When the top candidates are still very close, and we use the classifiers that are not confused between the candidates to further disambiguate the result. For real examples and further analysis, please refer to Section 5.3. Figure 4 presents this confusion matrix-guided ensemble from steps #30 to #49.

5. Experiments

Our experiments were designed to address the following questions:

- How do feature extraction algorithms compare with one another?
- Given an image category, which feature extraction algorithm performs the best, and why?
- What is the effect of the size of the training dataset?
- How does fusion perform compared with individual classifiers, and how do the results of fusion change with the number of training instances?

To answer the above questions, we conducted experiments on two datasets: an ImageNet dataset and a PicasaWeb dataset. The first dataset was constructed from ImageNet [12]. As our experiments aimed to study the effects that training dataset size has on feature extraction algorithms, we first selected 100 ImageNet categories (each with 1,100 images), and 10 PicasaWeb categories (each of which we manually annotate for a total of 11,000 images).

To make a fair comparison, feature vectors extracted from different algorithms were set to be the same length: 1,000. This was implemented by setting the

Table 3: Best category statistics.

Single Best	Count
CT	8
SIFT	29
HMAX	5
ConvNet	33

Tie for Best	Count
CT&SIFT	1
CT&ConvNet	5
SIFT&HMAX	7
SIFT&ConvNet	3
HMAX&ConvNet	9

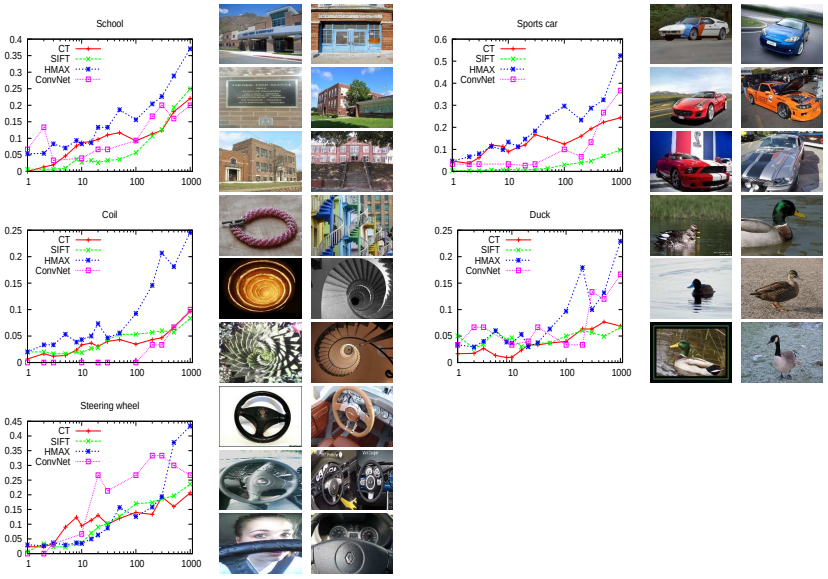

Figure 6: Good examples for HMAX. Categories include *school*, *sports car*, *coil*, *duck*, and *steering wheel*.

cluster number for color-texture and SIFT codebook, setting the patch number for HMAX, and setting the number of hidden units in the last layer for ConvNet. After extracting features for each image, the data were divided into training and testing sets for the supervised classifier. An 11-fold cross validation was performed. The data were randomly divided into 11 parts of equal size: 10 parts were used for training and the remaining one for testing. To evaluate the feature performance on different sizes of training instances, we also varied the training data from 1 to $1,000$ instances for each category ($10,000$ when available) and evaluated the accuracy using the same 100 testing data set aside for each fold. When we performed small experiments with only a part of the training data, a random subset was sampled. We used SVMs for the supervised learning, with PSVM [7] for large-scale data.

5.1. Results with ImageNet Dataset

Figure 5 shows the overall results for the ImageNet dataset. When comparing algorithms, each algorithm excels in categories with certain characteristics. It is not very meaningful to compare average classification accuracies on all categories because the composition of the testbed can be biased toward some algorithms. For the ImageNet dataset, SIFT always achieved much better results than other algorithms because the dataset contains a large number of categories favoring SIFT (We will provide more analysis later). Therefore, we did not analyze the exact values for overall accuracy, but only focused on the accuracy trends with respect to the growth of the training instance number. We make two observations. First, when there are only several training instances, it is insufficient to learn a practical model. No scheme can perform well. For example, when there is only one training instance for each category, all algorithms just perform slightly better than a random guess. However, as the simple algorithms, such as CT, do not require much data to learn model parameters, their results are relatively good when training data are sparse. For example, when the number of training instances ranges from 10 to 100, CT achieves better results than HMAX and ConvNet. Second, when the amount of training data increases, complex models such as HMAX and ConvNet can improve the results significantly. On large datasets, they achieve a higher rate of accuracy, and outperform CT when the number of training data increases to approximately 200 and above.

Because it is meaningless to only compare the average accuracy over all categories, we provide a comprehensive analysis to examine for which categories each algorithm provides the highest classification accuracy. The left part of Table 3 shows the number of winning categories of single algorithms, and the right part shows the tied winners (i.e., two methods achieve similar best results). ConvNet enjoys the most winning categories (33 out of 100), and SIFT ranks second with the ImageNet dataset. HMAX and ConvNet have nine common best categories, which indicates they have similar strengths with certain image categories because of their similar design. We also notice that there is no category where three or four algorithms all achieve the same best result at the same time. In the rest of this subsection, we list each feature extraction algorithm's advantages and present some illustrative image examples.

5.1.1. Color-Texture Codebook

Though simple, color-texture codebook enjoys the best performance for 11 categories. Figure 7 shows that these categories exhibit similar global color and/or texture distributions. The categories *school bus* and *bulldozer* both obviously have high yellow color percentages. *Fire extinguisher* has a good deal of red. *Penguin* exhibits a simple black-white pattern. *Laptop* displays uniform color on screens. *Eggs* have regular pure color or dots as texture. *Wine bottle* has constant green bottle with white caption. Category *car tire* exhibits an obvious pattern of tire texture.

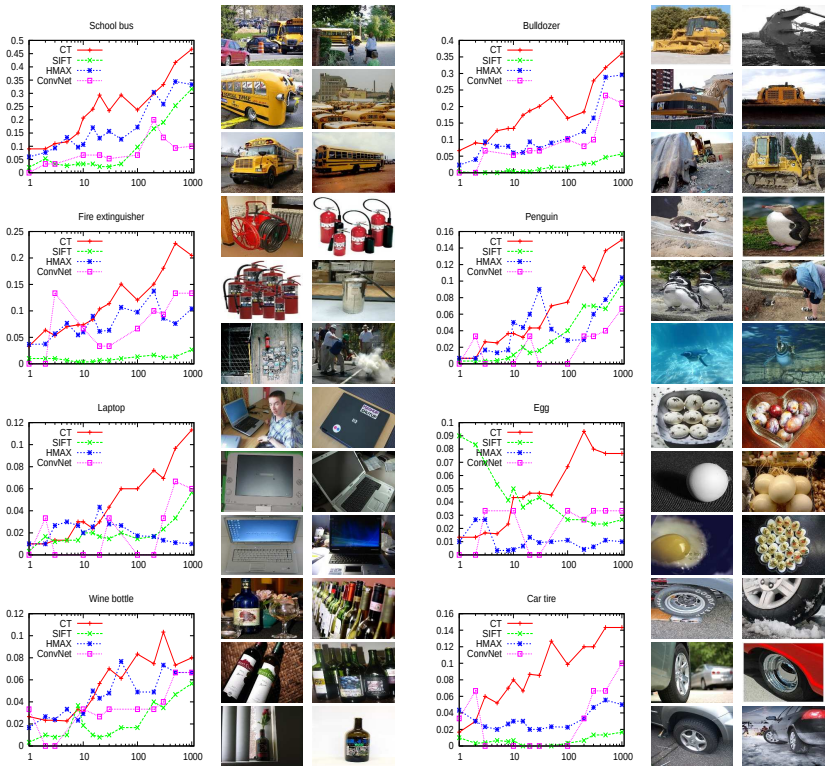

Figure 7: Good examples for CT. Categories include *school bus*, *bulldozer*, *fire extinguisher*, *penguin*, *laptop*, *egg*, *wine bottle*, and *car tire*.

5.1.2. SIFT Codebook

SIFT codebook focuses on characterizing local shapes. After detecting interesting points in an image, a 128-dimension SIFT descriptor is extracted for each interesting point. The SIFT descriptor is powerful for recognizing the local structures of objects. For example, a *ruler* has the narrow, rectangle-like ends and calibrations for length; both are regular local structures. These types of local structures cannot be detected by global matching or region statistics algorithms. Among the ImageNet dataset categories we constructed, 36 out of 100 categories favor SIFT for the local shape matching. These categories adhere to one of the following conditions.

Matching by Local Shapes. The first condition relates to objects with regular local shapes. These can be divided into two types: 1) when part of the object matches the whole object, and 2) when one object matches multiple objects of the same category. Successful categories for the first type include *ruler* and *calculator* (see Figure 8), where some images contain a whole ruler or calculator, and some contain only a part of the object. Similar conditions also exist for *chip* and *electric guitar*, which indicates significant local shape adhere to a local-to-whole mapping. For the second type, the successful category includes *pineapple*, where images have various numbers of units on pineapples. Similar conditions also exist for *strawberry*, *grape* and *deviled egg*, where the images have various numbers of the same object, which can only be solved by a local matching algorithm.

Matching by Near Duplication. The second condition is based on SIFT's strength in detecting near-duplicates. Although some images do not have obvious local structure, their positions of interesting points are stable, and their appearance based on local shapes is almost the same within categories. With enough data, similar images can be recognized in a near-duplicate detection way. Good example categories include *trilobite*, *palm*, *Bengal tiger*, *sea turtle*, *octopus*, *Dalmatian*, *tabloid*, and *pizza*, which are shown in Figure 9, where shapes of objects in one image can be quite similar to those in others, but may not be identical.

5.1.3. HMAX

HMAX extracts global patches. Its strength lies in its ability to identify whole objects instead of getting confused by focusing on their parts. Examples are shown in Figure 6. For example, *schools* must be recognized as a house with windows but not just windows. *Sports car* should have the whole shape with wheels instead of wheels only. The part-to-object feature extraction pipeline of the HMAX algorithm is a strength in its object detection method. HMAX outperforms SIFT in terms of recognizing objects as a whole than its parts.

The other strength of HMAX is that it focuses on dominant signals because of its *max* operation in its part selection step (see the HMAX description in Section 3.3). The *max* operation eliminates details in a patch. This allows HMAX

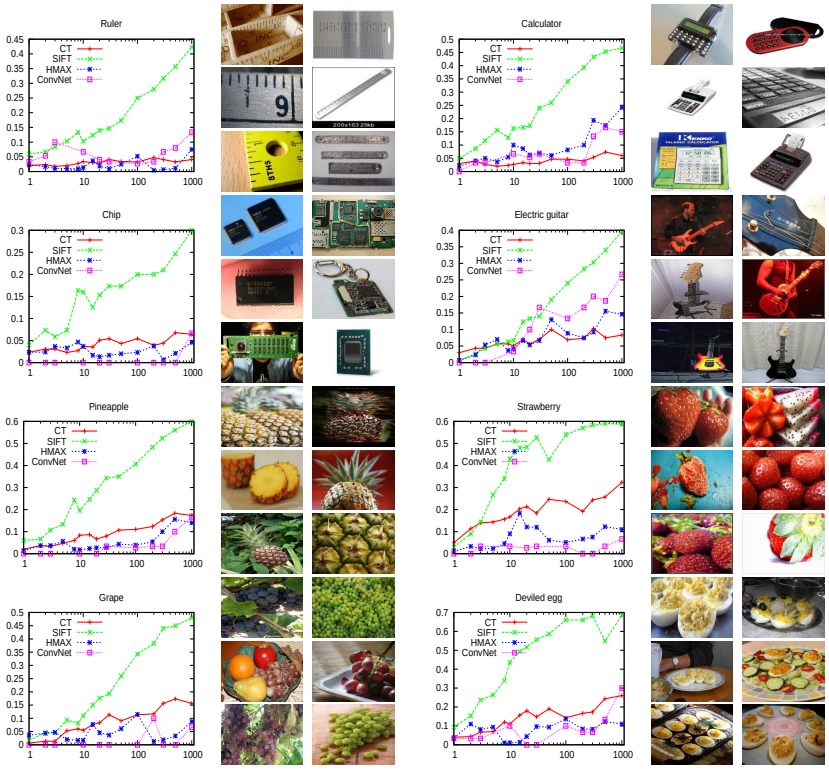

Figure 8: Good examples for SIFT: matching by local shapes. Categories include *ruler*, *calculator*, *chip*, *electric guitar*, *pineapple*, *strawberry*, *grape*, and *deviled egg*.

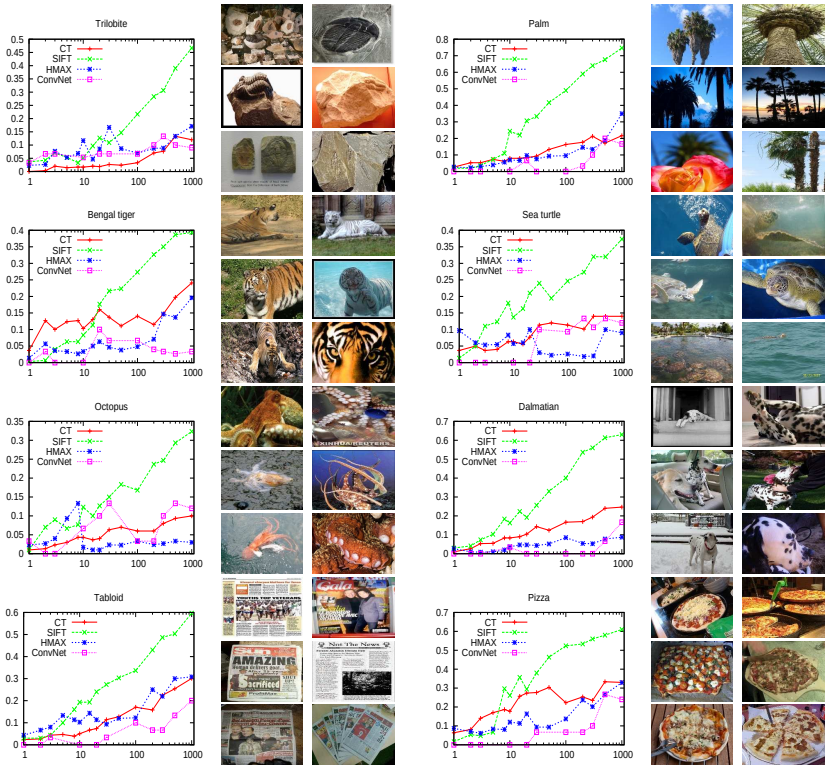

Figure 9: Good examples for SIFT: matching by near duplication. Categories include *trilobite*, *palm*, *bengal tiger*, *sea turtle*, *octopus*, *dalmatian*, *tabloid*, and *pizza*.

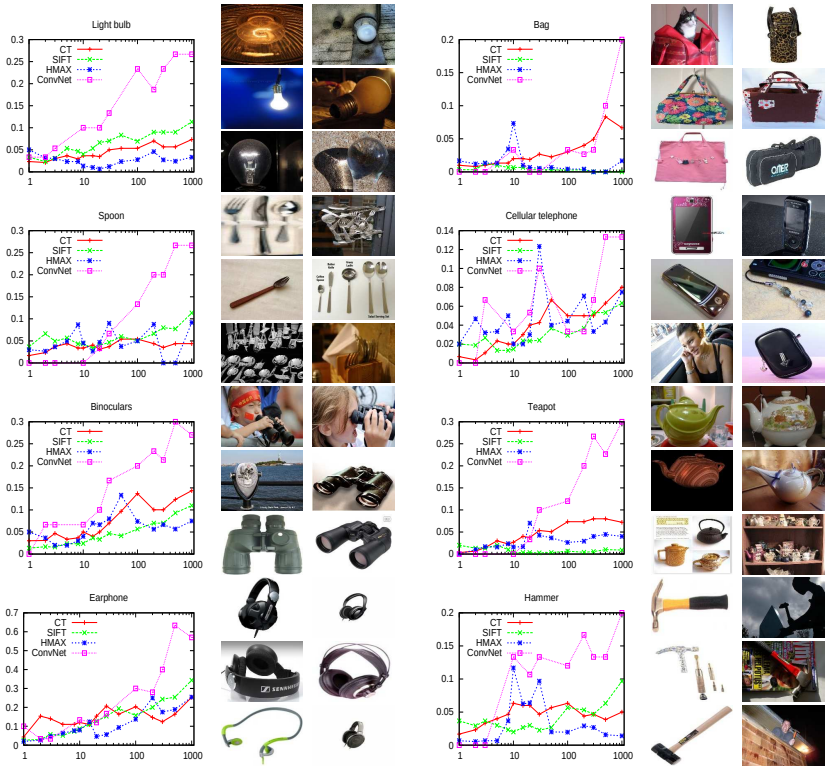

Figure 10: Good examples for ConvNet: patches with informative details. Categories include *light bulb*, *bag*, *spoon*, *cellular telephone*, *binoculars*, *teapot*, *earphone*, and *hammer*.

to outperform ConvNet when applied to objects that may have distracting details such as people wearing different clothes. Other examples are *coil*, *duck* and *steering wheel* where different details are recognized better by HMAX than by ConvNet.

5.1.4. ConvNet

ConvNet also relies on patches and focuses on global shape. Whereas HMAX focuses on characterizing objects, ConvNet also involves telling objects apart from one another. In other words, ConvNet was designed for classification. ConvNet learns patches carefully from training data through multiple iterations of convolution and sampling. Indeed, after so much more tuning, ConvNet wins the most number of categories when applied to the ImageNet dataset.

Patches with informative details. As ConvNet involves a deep learning pipeline to learn feature weighting parameters, it can pick up detailed information on

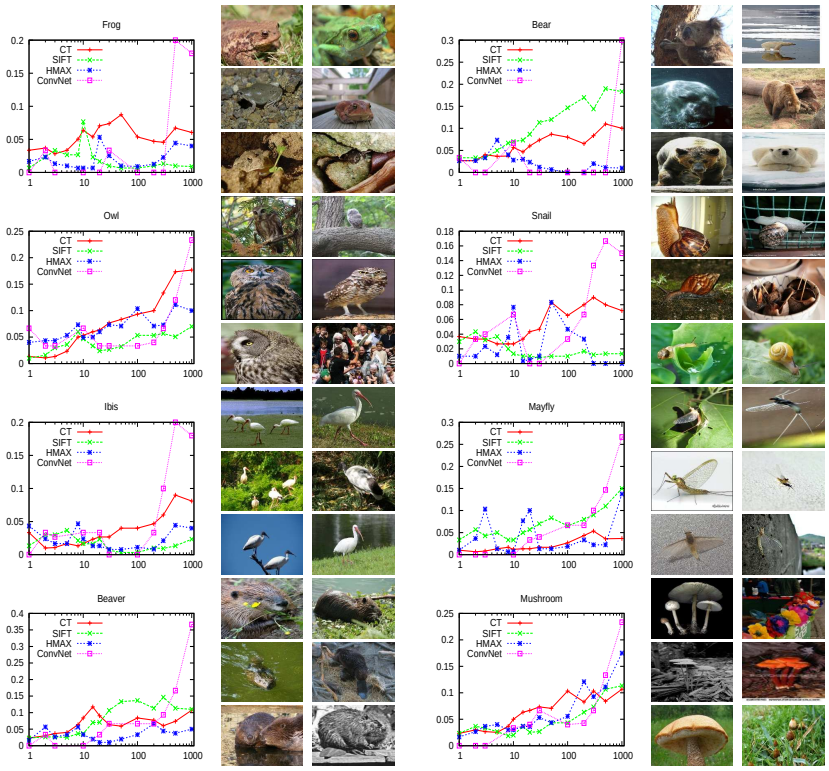

Figure 11: Good examples for ConvNet: patches for deformable objects. Categories include *frog*, *bear*, *owl*, *snail*, *ibis*, *mayfly*, *beaver*, and *mushroom*.

an object if those details may help telling objects apart. While enjoying better overall classification accuracy, it is understandable that ConvNet may overfit training data. One positive example for ConvNet is the *light bulb* (Figure 10), which has some regular shapes with various decorations. By just extracting the rounded top, the information may still be ambiguous in regards to telling a *light bulb* from a ping pong ball for example, so a set of well-tuned patches characterizing the interiors can be useful. The categories *bag* and *spoon* are other examples which have a rectangular shape with various patterns and orientations. The *cellular telephone* itself has a regular shape, and different models may have vastly different appearances. Though SIFT can perform well with *computer keyboard* images (see Figure 8), each of which has a similar appearance, SIFT does not do well with the category *cellular telephone*. Similar categories also include *binoculars*, *teapot*, *earphone*, and *hammer*. While shapes are important for classification, details are informative, and in some cases a learned set of patches can lead to more accurate results.

Patches for Deformable Objects. ConvNet performs well at identifying deformable objects such as the *frogs* and *bears* in Figure 11. With adequate training instances, ConvNet can capture an object at different orientations and different interior appearances. These living creatures do not have a regular local shape, so SIFT cannot detect interesting points well. HMAX does not perform as well as ConvNet when only parts of an object appear in an image, such as frogs' bodies or bears' heads. The random patch sampling algorithm in HMAX gives it less opportunity to capture object parts at different locations. These are reasons why ConvNet gives the best performance on deformable objects. Similar categories also include *owl*, *snail*, *ibis*, *mayfly*, *beaver*, and *mushroom*, which are all deformable objects solved by ConvNet when given a large-scale training dataset.

Notice that ConvNet often requires more training data than other schemes to start performing well. If we fix the number training instances as 100, ConvNet is often outperformed by HMAX or SIFT. When the number of training instances increases, ConvNet is able to recognize complex objects. We will further discuss the impact of training data size in Section 5.2.

5.2. Data-Driven Results with a Large Dataset

5.2.1. PicasaWeb Dataset Construction

Our goal was to construct a large-scale dataset with many images in each category. Therefore, we need a *huge number of images* and *frequent tags* for image collections. The procedure for constructing such a set includes the three parts.

Image Collection. To ensure a huge number of images, we collected 6.8 million of images from PicasaWeb with a "creative commons" license offered by users, which allows use and sharing for non-commercial purposes.

Tag Selection. To pool frequent tags, we applied a data-driven scheme. A total of 30,000 images were randomly selected from the dataset and annotated manually by 40 raters. Each rater explored each image to annotate as many tags as possible. These tags were then counted and sorted by frequency. The top tags were frequent tags for the selected images. After a manual filtering to remove meaningless ones, the tags are processed according to this order in the following steps.

Image Annotation for Each Tag. The most frequent tag was selected for annotation first. Among the correlated images with the tag from the annotated 30,000 preprocessing set, up to 500 images were selected as "seeds." Color histograms were extracted and applied in a k-nearest neighbor manner to sort all the remaining images in the 6.8 million image dataset. These images were scanned by the raters one by one to check whether it is a positive appearance of the corresponding tag. If we collected more than 11,000 images containing the tag, this tag was marked as "successful," and we moved on to the next tag with the same procedure. Otherwise, if 200,000 images were already scanned, but we still did not have 11,000 positive ones, we rejected the tag and moved on to the next one. Note that we assume that the negative appearances of tags always outnumber the positive ones; this is also the real case in practice. Therefore, we gathered 11,000 positive images and at least 11,000 negative images for each "successful" tag.

5.2.2. Experimental Results

For the purpose of this large-scale experiment, we collected 10 "successful" categories each with an excess of 11,000 labeled instances. These categories are as follows: *shoulder bag*, *horse/Equus caballus*, *baby/babe/infant*, *seashore/coast/seacoast/seacoast*, *city/metropolis*, *mountain/mount*, *afterglow*, *gravel/crushed rock*, *dog/domestic dog/Canis familiaris*, and *motorcycle/bike*. Figure 12 shows the average category-prediction accuracy, whereas Figure 13 presents the results of individual category examples. We can make three critical observations that are detailed as follows:

- *Data-Driven Works*

  When the training dataset size increases to 10,000 for each category, all algorithms converge to a similar level of average accuracy as in Figure 12. This illustrates why the fusion algorithm performs far better when the training data size is small but performs only a little better than individual methods when the training data size is large. Moreover, as shown in Figure 13, for most individual categories, such as *shoulder bag*, *city/metropolis*, and *dog/domestic dog/Canis familiaris*, all feature extraction algorithms reach almost the same level of class-prediction accuracy. It is surprising to see that the simplest model, CT, achieves comparable results when compared with other advanced models and applied to a large-scale PicasaWeb dataset. When the amount of training data is abundant,

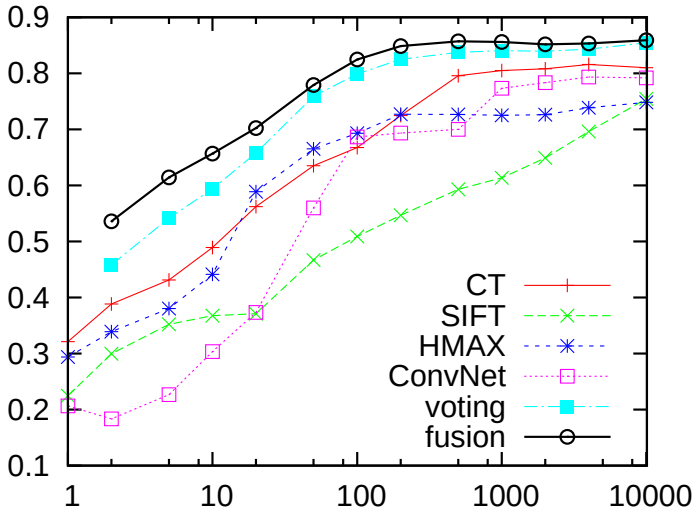

Figure 12: Accuracy for PicasaWeb dataset. X-axis shows the number of training instances for each category. Y-axis shows the average accuracy.

a simple model can also be competitive, which has also been confirmed for the text domain [55]. We also noticed that, for some categories, such as *mountain/mount* and *motorcycle/bike*, it appears that room for improvement exists for more training instances. Categories favoring each algorithm still exist, which agrees with the previous experiments.

- *Different Winners on Different Testbeds.*

  As we have illustrated through a per-category analysis, different algorithms may work better for different image categories. SIFT is the overall winner for the ImageNet dataset, whereas SIFT is almost the worst, and CT is the overall winner for the PicasaWeb dataset. In other words, two different datasets reach different conclusions when comparing feature extraction algorithms. This is because the ImageNet dataset contains more categories that are suitable for SIFT but the PicasaWeb dataset contains less. Results similar to this have often been used to justify one algorithm as being superior to another in the literature without noticing the dataset-composition limitation. Only a careful analysis using individual categories is able to provide sufficient justification to claim an algorithm to be superior to others.

- *Different Winners with Different Training-Size Values*

  As discussed in the previous section, the best algorithm for classifying a category can change with a different number of training instances. This observation was also demonstrated for the PicasaWeb dataset. As shown in Figure 12, when the number of training instances is approximately

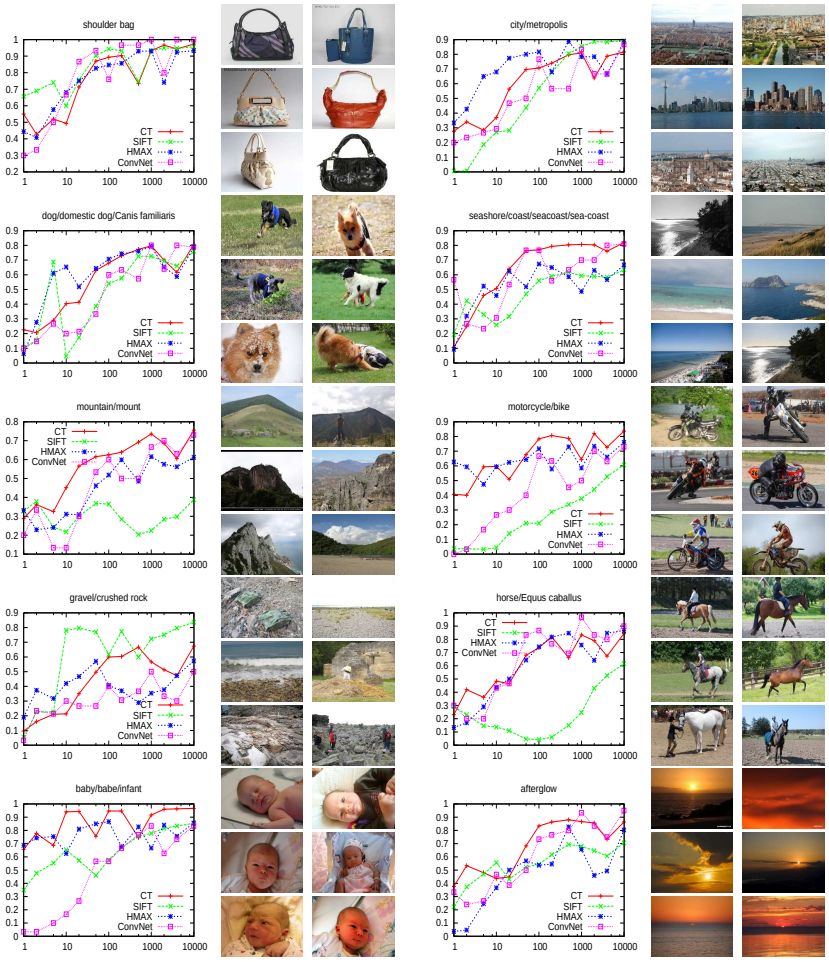

Figure 13: Examples for large data. Categories include *schoulder bag*, *city*, *dog*, *seashore*, *mountain*, *motocycle*, *gravel*, *horse*, *baby*, and *afterglow*.

10, CT is better than HMAX, and SIFT is better than ConvNet, which demonstrates that when the training data are sparse, the simple models may achieve better results than complex models. With the increase of the number of training instances, however, complex models, such as HMAX and ConvNet, can achieve better results. In Figure 12, when the number of instance increases beyond some level (approximately 20 instances on this dataset), the performance of HMAX and ConvNet improve. The above two descriptions were also demonstrated in the ImageNet dataset. Moreover, different algorithms may converge to similar good results when the training data are abundant. These results should warn those who want to compare algorithms that the comparison must be carefully performed on several sizes of training data to justify its conclusion. Combined with the second observation, both the testbed size and test-category composition can affect the conclusion of an algorithm-comparison.

5.3. Fusion Evaluation

To build a category-classifier relationship for fusion, we split the original training data randomly into 90% training and 10% validating (up to the nearest integer). We train individual classifiers with the training data and predict the validating data to construct confusion matrices in Section 5.3.1. Our fusion algorithm begins with at least two data instances because both training and validation sets require at least one data instance. The parameters for the fusion in Section 5.3.2 are as follows: gap threshold δ set to 0.05 and the weights $w_1 = 0.5$, $w_2 = 0.25$, $w_3 = 0.25$. This section shows the fusion results for the ImageNet dataset. Fusion results for the PicasaWeb dataset are shown together with the previous large data analysis.

5.3.1. Confusion Matrix

Previous sections discussed different performances with different categories. In fact, this was the inspiration for our fusion algorithm. By looking into the details of classification results, we constructed the confusion matrices in Figure 14, which reveal insights of feature differences among different feature extraction methods. The first four subfigures are confusion matrices for the four feature sets, CT, SIFT, HMAX, and ConvNet, respectively. The last subfigure is for our fusion results, which is discussed in the fusion section. We selected the highest values for non-diagonal positions until 10 categories were covered. Therefore, the categories in each figure look different. In the confusion matrix, each row corresponds to the ground truth label, whereas each column corresponds to the predicted label. Each value in the confusion matrix is between 0 and 1.

The *pizza* and *deviled egg* pair suffers from high confusion values for all methods (although HMAX has a slightly lower confusion value than the rest). This confusion arises because both categories have complex toppings with similar color, texture and shape. CT misclassifies many *tabloids* as belonging to the category *disc* because they both have various amounts of text on the surface.

Fortunately, this confusion does not happen through the "eyes" of the other three feature sets. SIFT finds it hard to distinguish *hammer* or *knife* from *pincer* for their similar local structure, but other feature sets easily address this ambiguity. CT may find their slightly different color and HMAX and ConvNet may find their global shape. HMAX appears to meet its bias condition for the categories *long trousers* and *spoon*. In this condition, when a classifier constructed by HMAX predicts a label *long trousers* or *spoon*, these consistently have low confidence levels. This confusion by HMAX can be substantially helped by using the other algorithms. ConvNet cannot distinguish the categories *garlic* and *deviled egg*. Other features can disambiguate this pair well however. For most of the confusion conditions, there is not much overlap on all feature sets. Our fusion algorithm is designed to use the trusted results and dismiss the confused results.

5.3.2. Fusion

As introduced in Section 4, the four classifiers are fused to construct a more powerful ensemble. For comparison, we also build a naive weighted-voting fusion algorithm. The voting scheme is quite simple: calculate the confidence for each category when validating data as the weights, and vote across the four features.

Refer again to Figure 5. Our fusion result performs better than both individual features alone and the weighted voting method. The key reason for this is our full usage of the confusion matrices.

Refer again to the confusion matrices presented in Figure 14. The figure at the bottom is the confusion matrix of the fusion scheme. Clearly, the confusion values in that matrix are smaller than those in the confusion matrices of the individual classifiers. Specifically, for the *pizza* and *deviled egg* pair, their confusion value is still high in the fused matrix but less than that in either of the individual matrices. This example indicates that our fusion scheme reduces confusion and thus improves classification accuracy.

Let us use an *hammer* example to explain how the fusion scheme helps. The SIFT algorithm returns *pincer*. The CT algorithm returns *sextant* as Figure 14 shows. The HMAX algorithm returns *umbrella*, and the ConvNet algorithm returns *earphone*. None of these results is correct, but we can extend them for fusion. The SIFT algorithm extends the candidates to *hammer*, *knife*, and *pincer* with a coverage of 81%. The CT algorithm extends the candidates to *hat*, *sextant*, *deviled egg*, and *pineapple* with a coverage of 33%. The HMAX algorithm extends the candidates to *umbrella*, *skeleton key*, and *ruler* with a coverage of 5.2%. Lastly, the ConvNet algorithm extends the candidates to *earphone*, *spectacles*, and *steering wheel* with a coverage of 38.1%. So the highest coverage SIFT is selected for the first classifier. *Hammer*, *knife* and *pincer* are voted on by the remaining three algorithms. In this example, *hammer* is voted with the highest confidence, and so the fusion algorithm predicts the class as *hammer*. No individual features provide a correct result on this case, but a fusion algorithm can mine correlations between categories and take advantage

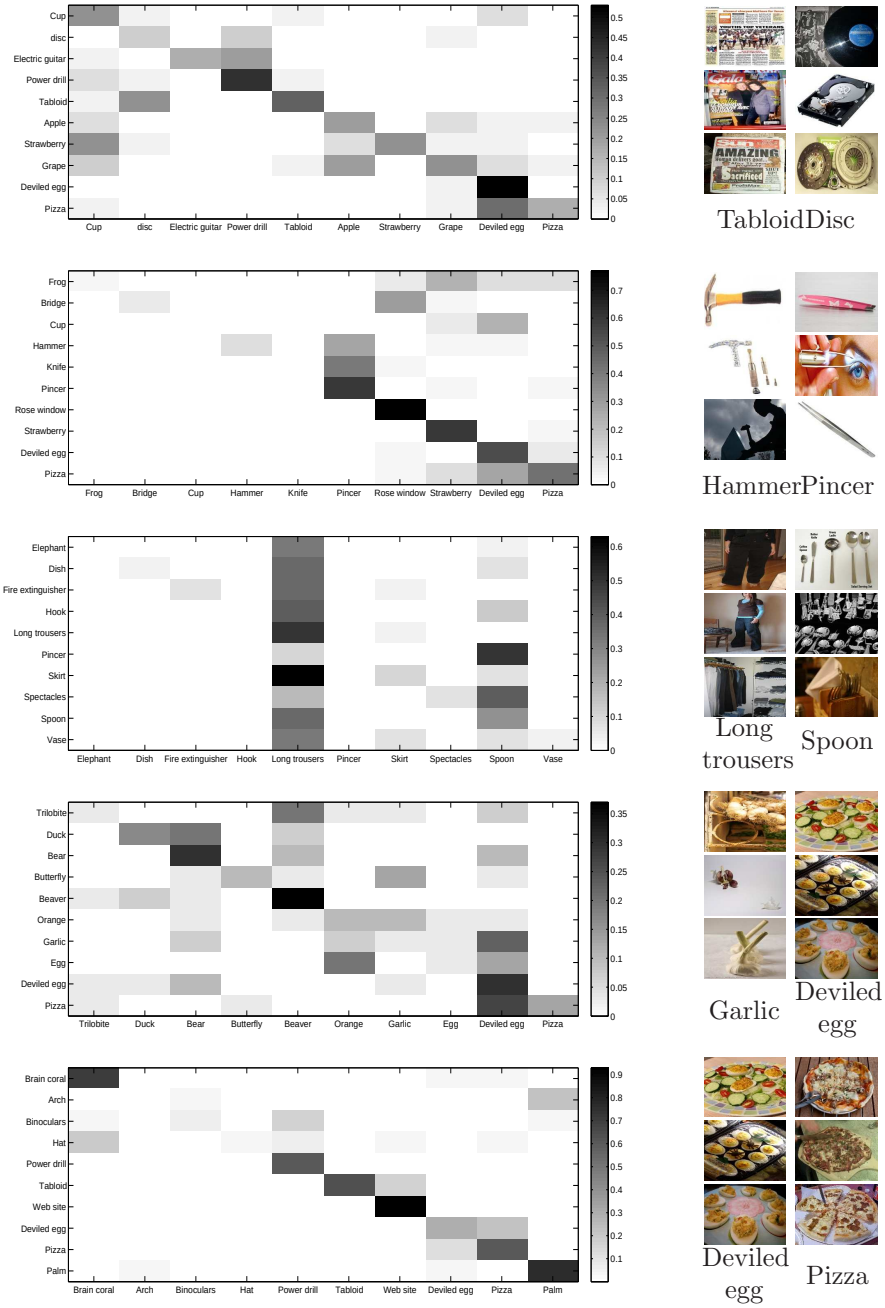

Figure 14: Confusion matrix for CT, SIFT, HMAX, ConvNet, and fusion (top to down). Rows stand for ground truth labels, and columns stand for predicted labels. A darker color indicates a larger value in the confusion matrix.

of multiple feature-extraction algorithms to obtain a better chance of getting a correct result.

We also observe that when the number of training instances is small, the fusion approach significantly increases the accuracy. However, when the number of training increases, the gap between the fusion and individual methods decreases. The reason is that when the number of training instances is small, each feature is insufficient for learning a great deal of information, which leads to more confusion. With a fusion algorithm, the confusion stemming from lack of information is greatly decreased. When the number of training instances increases, however, confusion of this sort decreases, and, accordingly, so does the gap.

6. Conclusion

In this paper, we investigated four representative feature extraction algorithms, color-texture codebook (CT), SIFT codebook, HMAX, and convolutional networks (ConvNet). Comprehensive experiments were conducted that revealed differences between these algorithms. We discussed our results using two different views. The first view is the image-category view. We provided an extensive analysis of different categories, and found that different algorithms each have their own advantages that can give them an edge in different categories. This finding is one of the main reasons why different algorithms may achieve significantly different or even contradictory results with different datasets. In addition to the inter-category analysis, this actually reveals patterns of intra-category variance. Different feature extraction algorithms discover different intra-category invariant information, and thus perform differently. The second view is the number of training instances. We observed that different algorithms may perform differently depending on the number of training instances. Simple algorithms tend to perform better when the training data size is small. When the number of training instances reaches up to and beyond a certain amount, complex models can experience a sudden jump in accuracy. To analyze even larger number of training instances, we constructed a new large-scale image dataset from PicasaWeb. With this dataset, we observed that all the four algorithms we studied converge to a good accuracy level with an abundance of training instances. The observation demonstrated the validity of the data-driven approach. Our experimental results reveal that both training data size and dataset category composition can affect the results of algorithm comparisons. Extreme care must be taken by researchers to avoid such pitfalls to ensure the reporting of reliable results.

Finally, we devised and studied a fusion algorithm based on confusion matrices to harvest synergies between these four algorithms. The key idea is that when an algorithm is confused between classes x and y for recognizing an object, one should direct the recognition of the object to an algorithm that can clearly distinguish the two classes. Our fusion method can improve class-prediction

accuracy by substantial margins when the training dataset size ranges from small to moderate (below $1,000$). When the dataset size is large (approaching $5,000$), improvement still exists, although it is less significant. Fusion methods that can take advantage of having different views of the raw data can continue to improve class-prediction accuracy even when individual views have reached their limits.

We believe that our proposed confusion matrix-guided fusion scheme can be formulated into an optimization problem to obtain statically optimal class prediction. Our future work will be devoted into formally formulating the problem and comparing different design considerations.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (No. 61370022, No. 61003097, No. 60933013, and No. 61210008), International Science and Technology Cooperation Program of China (No. 2013DFG12870), and the National Program on Key Basic Research Project (No. 2011CB302206).

References

[1] Naif Alajlan, Yakoub Bazi, Farid Melgani, and Ronald R Yager. Fusion of supervised and unsupervised learning for improved classification of hyperspectral images. *Information Sciences*, 2012.

[2] Y. Bengio. *Learning deep architectures for AI.* Now Publishers Inc, 2009.

[3] Y.L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2559–2566. IEEE, 2010.

[4] Y. Cao, C. Wang, Z. Li, L. Zhang, and L. Zhang. Spatial-bag-of-features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[5] E.Y. Chang. *Foundations of Large-Scale Multimedia Information Management and Retrieval: Mathematics of Perception.* Springer-Verlag New York Inc, 2011.

[6] E.Y. Chang, B. Li, and C. Li. Toward perception-based image retrieval. In *Content-based Access of Image and Video Libraries, 2000. Proceedings. IEEE Workshop on*, pages 101–105. IEEE, 2000.

[7] E.Y. Chang, K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, and H. Cui. Psvm: Parallelizing support vector machines on distributed computers. *Advances in Neural Information Processing Systems*, 20:16, 2007.

[8] T. Chang and C.C.J. Kuo. Texture analysis and classification with tree-structured wavelet transform. *Image Processing*, 2(4):429–441, 1993.

[9] Peng Cui, Zhi-Qiang Liu, Li-Feng Sun, and Shi-Qiang Yang. Hierarchical visual event pattern mining and its applications. *Data Mining and Knowledge Discovery*, 22(3):467–492, 2011.

[10] Peng Cui, Li-Feng Sun, Zhi-Qiang Liu, and Shi-Qiang Yang. A sequential monte carlo approach to anomaly detection in tracking visual events. In *Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[11] Peng Cui, Fei Wang, Li-Feng Sun, Jian-Wei Zhang, and Shi-Qiang Yang. A matrix-based approach to unsupervised human action categorization. *IEEE Transactions on Multimedia*, 14(1):102–110, 2012.

[12] J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, and L. Fei-Fei. ImageNet: a large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[13] T. Deselaers, D. Keysers, and H. Ney. Features for image retrieval: an experimental comparison. *Information Retrieval*, 11(2):77–107, 2008.

[14] Ofer Dor and Yoram Reich. Strengthening learning algorithms by feature discovery. *Information Sciences*, 189:176–190, 2012.

[15] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[16] T.J. Gawne and J.M. Martin. Responses of primate visual cortical V4 neurons to simultaneously presented stimuli. *Journal of Neurophysiology*, 88(3):1128, 2002.

[17] J.M. Geusebroek, R. van den Boomgaard, A.W.M. Smeulders, and H. Geerts. Color invariance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1338–1350, 2001.

[18] T. Gevers, WM Smeulders, et al. Color based object recognition. *Pattern recognition*, 32(3):453–464, 1999.

[19] T. Gevers and H. Stokman. Robust histogram construction from color invariants for object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):113–118, 2004.

[20] Abby A Goodrum. Image information retrieval: An overview of current research. *Informing Science*, 3(2):63–66, 2000.

[21] G.E. Hinton, S. Osindero, and Y.W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[22] DH Hubel and TN Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, 195(1):215, 1968.

[23] Giridharan Iyengar and Harriet J Nock. Discriminative model fusion for semantic concept detection and annotation in video. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 255–258. ACM, 2003.

[24] H. Jégou, M. Douze, and C. Schmid. Packing bag-of-features. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[25] Y.G. Jiang, C.W. Ngo, and J. Yang. Towards optimal bag-of-features for object categorization and semantic video retrieval. In *ACM international conference on Image and video retrieval*, 2007.

[26] R. Kohavi and F. Provost. Glossary of terms. *Machine Learning*, 30(June):271–274, 1998.

[27] I. Lampl, D. Ferster, T. Poggio, and M. Riesenhuber. Intracellular measurements of spatial integration and the MAX operation in complex cells of the cat primary visual cortex. *Journal of Neurophysiology*, 92(5):2704, 2004.

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[29] J.G. Leu. Computing a shape's moments from its boundary. *Pattern Recognition*, 24(10):949–957, 1991.

[30] Shaowei Liu, Peng Cui, Huanbo Luan, Wenwu Zhu, Shiqiang Yang, and Qi Tian. Social-oriented visual image search. *Computer Vision and Image Understanding*, 2013.

[31] D.G. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision (ICCV)*, 1999.

[32] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[33] E.K. Miller. The prefrontal cortex and cognitive control. *Nature Reviews Neuroscience*, 1(1):59–66, 2000.

[34] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[35] A. Oliva and P.G. Schyns. Coarse blobs or fine edges? evidence that information diagnosticity changes the perception of complex visual stimuli. *Cognitive psychology*, 34:72–107, 1997.

[36] Daniel Carlos Guimarães Pedronette and Ricardo da S Torres. Exploiting pairwise recommendation and clustering strategies for image re-ranking. *Information Sciences*, 207:19–34, 2012.

[37] O. Pichler, A. Teuner, and B.J. Hosticka. A comparison of texture feature extraction using adaptive gabor filtering, pyramidal and tree structured wavelet transforms. *Pattern Recognition*, 29(5):733–742, 1996.

[38] S. Prasad and L.M. Bruce. Decision fusion with confidence-based weight assignment for hyperspectral target recognition. *Geoscience and Remote Sensing*, 46(5):1448–1456, 2008.

[39] MA Ranzato, F.J. Huang, Y.L. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[40] M. Riesenhuber and T. Poggio. Are Cortical Models Really Bound by the Binding Problem. *Neuron*, 24(1):87–93, 1999.

[41] J.J. Rodriguez, L.I. Kuncheva, and C.J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.

[42] J.A. Roubos, M. Setnes, and J. Abonyi. Learning fuzzy classification rules from labeled data. *Information Sciences*, 150(1):77–93, 2003.

[43] Pierre Sermanet and Yann LeCun. Traffic Sign Recognition with Multi-Scale Convolutional Networks. In *IJCNN*, 2011.

[44] T. Serre. *Learning a dictionary of shape-components in visual cortex: comparison with neurons, humans and machines.* PhD thesis, Massachusetts Institute of Technology, 2006.

[45] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):411–426, 2007.

[46] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[47] J. Smith and S.F. Chang. Automated image retrieval using color and texture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1996.

[48] Cees GM Snoek, Marcel Worring, and Arnold WM Smeulders. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 399–402. ACM, 2005.

[49] Q.S. Sun, S.G. Zeng, Y. Liu, P.A. Heng, and D.S. Xia. A new method of feature fusion and its application in image recognition. *Pattern Recognition*, 38(12):2437–2448, 2005.

[50] S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *ACM international conference on Multimedia*, pages 107–118, 2001.

[51] J. Van De Weijer, T. Gevers, and A.D. Bagdanov. Boosting color saliency in image feature detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):150–156, 2006.

[52] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 606–613. IEEE, 2009.

[53] Meng Wang, Xian-Sheng Hua, Richang Hong, Jinhui Tang, Guo-Jun Qi, and Yan Song. Unified video annotation via multigraph learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(5):733–746, 2009.

[54] Meng Wang, Hao Li, Dacheng Tao, Ke Lu, and Xindong Wu. Multimodal graph-based reranking for web image search. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, 21(11):4649, 2012.

[55] C. Xu, K. Chiew, Y. Chen, and J. Liu. Fusion of text and image features: A new approach to image spam filtering. *Practical Applications of Intelligent Systems*, pages 129–140, 2012.

[56] Ronald R Yager. A framework for multi-source data fusion. *Information Sciences*, 163(1):175–200, 2004.

[57] J. Yang, J. Yang, D. Zhang, and J. Lu. Feature fusion: parallel strategy vs. serial strategy. *Pattern Recognition*, 36(6):1369–1381, 2003.

[58] M. Yasuda, T. Banno, and H. Komatsu. Color selectivity of neurons in the posterior inferior temporal cortex of the macaque monkey. *Cerebral Cortex*, 2009.

[59] Zheng-Jun Zha, Xian-Sheng Hua, Tao Mei, Jingdong Wang, Guo-Jun Qi, and Zengfu Wang. Joint multi-label multi-instance learning for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[60] Zheng-Jun Zha, Tao Mei, Jingdong Wang, Zengfu Wang, and Xian-Sheng Hua. Graph-based semi-supervised learning with multiple labels. *Journal of Visual Communication and Image Representation*, 20(2):97–103, 2009.

[61] Zheng-Jun Zha, Meng Wang, Yan-Tao Zheng, Yi Yang, Richang Hong, and Tat-Seng Chua. Interactive video indexing with statistical active learning. *IEEE Transactions on Multimedia*, 14(1):17–27, 2012.

[62] Zheng-Jun Zha, Linjun Yang, Tao Mei, Meng Wang, and Zengfu Wang. Visual query suggestion. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 15–24. ACM, 2009.

[63] X. Zhang, L. Zhang, XJ Wang, and H. Shum. Finding celebrities in billions of web images. *IEEE Transactions on Multimedia*, (99):1–1, 2011.