Representing Sets of Instances for Visual Recognition

Jianxin Wu,[1] **Bin-Bin Gao,**[1] **Guoqing Liu**[2]
[1] National Key Laboratory for Novel Software Technology
Nanjing University, China
[*] wujx2001@nju.edu.cn, gaobb@lamda.nju.edu.cn
[2] Minieye, Youjia Innovation LLC, China
guoqing@minieye.cc

Abstract

In computer vision, a complex entity such as an image or video is often represented as a set of instance vectors, which are extracted from different parts of that entity. Thus, it is essential to design a representation to encode information in a set of instances robustly. Existing methods such as FV and VLAD are designed based on a generative perspective, and their performances fluctuate when difference types of instance vectors are used (i.e., they are not robust). The proposed D3 method effectively compares two sets as two distributions, and proposes a directional total variation distance (DTVD) to measure their dissimilarity. Furthermore, a robust classifier-based method is proposed to estimate DTVD robustly, and to efficiently represent these sets. D3 is evaluated in action and image recognition tasks. It achieves excellent robustness, accuracy and speed.

Introduction

In visual recognition, a complex entity (*e.g.*, image or video) is usually represented as a set of instance vectors. Each instance vector is extracted using part of the entity (*e.g.*, a local window extracted from an image or a time-space subvolume from a video). Various features have been used to extract instance vectors, such as dense SIFT (Lowe 2004) and CNN features for images (Jia et al. 2014), or improved dense trajectory features (Wang and Schmid 2013) for videos (Gkioxari and Malik 2015). Recent works have shown that if a set of CNN features are extracted from entities and classify images or videos based on these sets, higher accuracy can be obtained (Gong et al. 2014; Cimpoi, Maji, and Vedaldi 2015; Xu, Yang, and Hauptmann 2015).

Because most existing learning algorithms assume that an entity is represented as a single vector instead of a set of vectors, we need to find a suitable visual representation that encodes the set of instance vectors into one single vector. It is desirable that the representation will capture useful (*i.e.*, discriminative) information from the set.

Fisher Vector (FV) and VLAD are the most widely used representations for this task. FV (Sánchez et al. 2013) is based on the idea of Fisher kernel in machine learning. It models the distribution of instance vectors in training entities using a Gaussian Mixture Model (GMM). Then, one training or testing entity is modeled generatively, by a vector which describes how the GMM can be modified to generate the instance vectors inside that entity. A GMM with K components has three sets of parameters $(w_i, \boldsymbol{\mu}_i, \Sigma_i)$, $1 \le i \le K$. VLAD (Jégou et al. 2012), another popular visual representation, can be regarded as a special case of FV, by using only the $\boldsymbol{\mu}$ parameters. The classic bag-of-visual-words (BOVW) (Csurka et al. 2004) representation is also a special case of FV, using only the w parameters.

However, FV and VLAD share two major limitations. First, recognition performance based on them are *not robust*. For some tasks, FV may have high accuracy but VLAD has poor performance, while the trend may be reversed in other tasks (cf. the experimental results section). Since we do not know *a priori* which representation (FV or VLAD) is better for a specific task, we prefer a novel representation which is robust (*i.e.*, always has high accuracy) in all tasks. Second, they both focus on modeling how one entity or one distribution is *generated*. Given the fact that the task in hand is recognition, we argue that *we need to pay more attention to how two entities or two distributions are separated.* In other words, we need a visual representation that pays more attention to the discriminative side.

In this paper, we propose a discriminative distribution distance (D3) representation that converts a set of instance vectors into a vector representation. D3 explicitly considers two distributions: a density T which is estimated from the training set as a reference model, and one entity forms another distribution X. D3 then uses the distribution distance between T and X as a discriminative representation for the entity X. Technically, D3 has the following contributions.

- We propose DTVD, a directional total variation distance, to measure the distance between two distributions, which contains more discriminative information than classic distribution distances by considering the *direction*;
- Directly calculating DTVD is problematic because T or X may be non-Gaussian or only contains few items. We propose to estimate DTVD in a discriminative manner, by *calculating* robust *misclassification rates* when we try to classify every dimension of T from X;
- We show by experiments that D3 has *robustly* achieved high accuracy. We also show that D3 and FV are com-

[*]This research was supported by the National Natural Science Foundation of China under Grant 61422203.

plementary to each other. By combining D3 and FV, we achieve the highest accuracy.

Background: the FV representation

In the Fisher Vector (FV) representation, a large set of instance vectors are extracted from training images or videos. We treat this set as T and a Gaussian Mixture Model (GMM) p with parameters $\lambda = (w_k, \boldsymbol{\mu}_k, \Sigma_k), 1 \leq k \leq K$, is estimated from T. When a test image or video is presented, we extract its instance vectors to form a set X. The FV representation encodes X as a vector $\boldsymbol{x}$. This is a generative model, and each component in the vector representation $\boldsymbol{x}$ describes how each parameter in λ should be modified such that p can be tuned to fit data X properly. Given two entities X and Y, the distance between them can be calculated using the distance between the encoded representation $\boldsymbol{x}$ and $\boldsymbol{y}$, a fact that leads to high computational efficiency.[1]

Thus, the key issue is: given the set T (training entities) and any entity X, how shall we generate $\boldsymbol{x}$ such that $\boldsymbol{x}$ encodes important information that distinguishes X from T?

We have the following observations based on FV.

- The vector $\boldsymbol{x}$ in FV is formed under the *generative* assumption that X can be generated by p if we are allowed to modify the parameter set λ. Since what we are really interested in is how different is X from T, we believe that a *discriminative* distance between X and T is a better option. That is, in this paper we will treat X and T as sampled from two different distributions, and find their distribution *distance* to encode the image or video X. A representation of X that encodes the distance between X and and the reference model T will contain useful discriminative information about X;
- Diagonal Σ_k and linear classifiers are used in FV. Thus, it is reasonable to approximately treat each dimension of $\boldsymbol{x}$ as independent of other dimensions. Thus, in finding a suitable representation for X, we can consider every dimension individually. That is, given two sets of *scalar* values $X = \{x_1, x_2, \ldots, \}$ and $Y = \{y_1, y_2, \ldots\}$ (sampled from 1-d distribution p_X and p_Y, respectively), how do we properly compute the distance $d(p_X, p_Y)$?[2]

Discriminative Distribution Distance

Directional Total Variation Distance

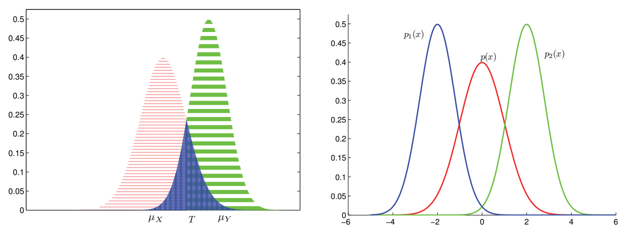
A widely used distance that compares two distributions is the *total variation* distance, which is independent of the distributions' parameterizations. Let ν_X and ν_Y be two probability measures on a measurable space $(\mathcal{O}, \mathcal{B})$, the total variation distance is defined as

$$d_{TV}(\nu_X, \nu_Y) \triangleq \sup_{A \in \mathcal{B}} |\nu_X(A) - \nu_Y(A)| \,. \qquad (1)$$

While this definition is rather formal, d_{TV} has a more intuitive equation for commonly used continuous distributions

[1]The details of VLAD are omitted, because VLAD can be considered as a special case of FV.

[2]We use X and Y to represent two distributions, instead of confined to T and X in the FV context, because the DTVD distance proposed in this paper is more general.



(a) (b)

Figure 1: Illustration of the total variation distance. 1a illustrates d_{TV} for two Gaussians, and 1b reveals that *direction* is essential to distinguish p_1 from p_2.

by the Scheffe's Lemma (DasGupta 2011). For example, for two normal distributions with p.d.f. $p_X = N(\mu_X, \sigma_X^2)$ and $p_Y = N(\mu_Y, \sigma_Y^2)$,

$$d_{TV}(p_X, p_Y) = \frac{1}{2} \int_u |p_X(u) - p_Y(u)| \, \mathrm{d}u \,. \qquad (2)$$

As illustrated in Fig. 1a, it is half the summed area of the red and green regions, which clearly indicates how two distributions are separated from each other.

The classic total variation distance (Eq. 2), however, is missing one most important information that captures the key difference between p_X and p_Y, as shown in Fig. 1b. In Fig. 1b, p_1 and p_2 are symmetric with respect to the mean of p, thus we have $d_{TV}(p, p_1) = d_{TV}(p, p_2)$, in spite of the fact that p_1 and p_2 are far apart. The missing of *direction* information is responsible for this drawback. Thus, we propose a directional total variation distance (DTVD) as

$$d_{DTV}(p_X, p_Y) = \mathrm{sign}(\mu_Y - \mu_X) \times d_{TV}(p_X, p_Y) \,. \qquad (3)$$

DTVD is a signed distance. In Fig. 1b, we will (correctly) have $d_{DTV}(p, p_1) = -d_{DTV}(p, p_2)$, which clearly signifies that p_1 and p_2 are on different sides (directions) of p and hence they are far from each other.

Robust estimation of the DTVD

For two Gaussians p_X and p_Y, their p.d.f. will have two intersections if $\sigma_X \neq \sigma_Y$. For example, in Fig. 1a the second intersection is in the far right end of the x-axis. A closed-form solution to calculate d_{TV} based on both intersections is available (DasGupta 2011). However, this closed-form solution leads to serious performance drop when used in visual recognition in our experiments. We conjecture that two reasons have caused this issue: The distributions are not necessarily normal and the estimation of distribution parameters are *not robust*. The shape of distributions resemble that of a Gaussian, but still have obvious deviations. For one distribution estimated from a single entity (cf. the right figure in Fig. 2b), it usually contains small number of instance vectors, which leads to unstable estimation of its distribution parameters, and hence unstable $d_{DTV}(p_X, p_Y)$. Thus, we need a more *robust* way to estimate the distribution distance.

Our key insight again arises from the discriminative perspective. It is obvious that the total variation distance d_{TV}

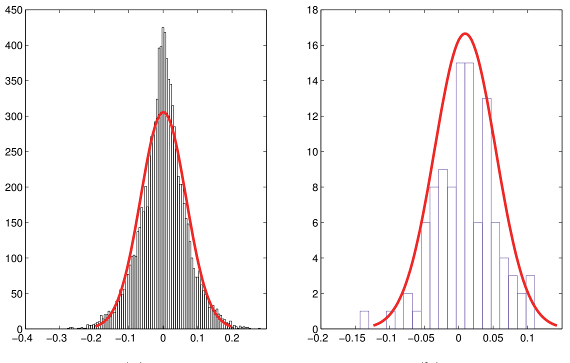


Figure 2: Typical distribution of feature values. 2a is calculated based on all training entities, and 2b is from a single entity. The red curve is a normal distribution estimated from the same data. This figure is generated using bag of dense SIFT on the Scene 15 dataset with $K = 64$ VLAD encoding. The dimension shown is the 37-th dimension in the 37-th cluster of the codebook.

is equivalent to one minus the *Bayes error* of a binary classification problem, where the two classes have equal prior and follow p_X and p_Y, respectively. Thus, we can estimate d_{TV} (hence d_{DTV}) by *robustly estimating the classification error between the two sets of examples X and Y*. Note that this task is easy since X and Y only contain scalar examples.

A classifier is *robust* means that it can achieve good (or at least reasonable) accuracy when its assumptions are not precisely satisfied (*e.g.*, the distributions are not Gaussian and the estimation of distribution parameters are not very reliable). We adopt the minimax probability machine (MPM) (Lanckriet et al. 2002) to robustly estimate the classification error. MPM is robust because its objective is to minimize the maximum probability of misclassification (*i.e.*, to ensure the smallest error in the worst case scenario).

Given examples X with mean $\boldsymbol{\mu}_X$ and covariance Σ_X and examples Y with $\boldsymbol{\mu}_Y$ and Σ_Y, the optimal MPM boundary $\boldsymbol{a}_\star^T \boldsymbol{x} - b_\star = 0$ is determined as (Lanckriet et al. 2002)[3]

$$\kappa_\star^{-1} \triangleq \min_{\boldsymbol{a}} \sqrt{\boldsymbol{a}^T \Sigma_X \boldsymbol{a}} + \sqrt{\boldsymbol{a}^T \Sigma_Y \boldsymbol{a}} \tag{4}$$

$$\text{s.t.} \quad \boldsymbol{a}^T (\boldsymbol{\mu}_X - \boldsymbol{\mu}_Y) = 1 \,, \tag{5}$$

and

$$b_\star = \boldsymbol{a}_\star^T \boldsymbol{\mu}_X - \kappa_\star \sqrt{\boldsymbol{a}_\star^T \Sigma_X \boldsymbol{a}_\star} \,, \tag{6}$$

where $\boldsymbol{a}_\star$ is an optimal solution of Eq. 5.

This is a second order cone problem (SOCP) that can be solved by an iterative algorithm. However, since we are dealing with scalar examples that (assumed to approximately) follow normal distributions, it has a closed form solution. Note that $X \sim N(\mu_X, \sigma_X^2)$ and $Y \sim N(\mu_Y, \sigma_Y^2)$, Eq. 5 reduces to $a(\mu_X - \mu_Y) = 1$, hence $a = \frac{1}{\mu_X - \mu_Y}$. Similarly, $\boldsymbol{a}^T \Sigma_X \boldsymbol{a} = (a\sigma_X)^2$, and Eq. 4 reduces to $\kappa_\star^{-1} = |a|(\sigma_X + \sigma_Y)$. Putting these results together and using Eq. 6, we get (cf. the supplementary material for details)

$$a_\star = \frac{1}{\mu_X - \mu_Y} \,, \quad b_\star = a_\star \times \frac{\mu_X \sigma_Y + \mu_Y \sigma_X}{\sigma_X + \sigma_Y} \,, \tag{7}$$

$$\kappa_\star = \frac{|\mu_X - \mu_Y|}{\sigma_X + \sigma_Y} \,. \tag{8}$$

The boundary $ax - b$ is equivalent to $x - \frac{b}{a}$. Hence, the two 1-d distributions p_X and p_Y are separated at the threshold value

$$T = \frac{b_\star}{a_\star} = \frac{\mu_X \sigma_Y + \mu_Y \sigma_X}{\sigma_X + \sigma_Y} \,. \tag{9}$$

If we re-use Fig. 1a and (approximately) assume the red, blue, and green areas intersect at $T = \frac{\mu_X \sigma_Y + \mu_Y \sigma_X}{\sigma_X + \sigma_Y}$, which is guaranteed to reside in between μ_X and μ_Y. Then, the area of the blue region is:[4]

$$Area = 1 - \Phi\left(\frac{T - \mu_X}{\sigma_X}\right) + \Phi\left(\frac{T - \mu_Y}{\sigma_Y}\right) \,. \tag{10}$$

where

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} \, \mathrm{d}t \tag{11}$$

is the cumulative distribution function (c.d.f.) of a standard normal distribution $N(0, 1)$. And, we have

$$d_{DTV}(p_X, p_Y) = \frac{1}{2}(2 - 2Area) = 2\Phi\left(\frac{\mu_Y - \mu_X}{\sigma_X + \sigma_Y}\right) - 1 \,, \tag{12}$$

making use of the fact that

$$\frac{T - \mu_X}{\sigma_X} = \frac{\mu_Y - \mu_X}{\sigma_X + \sigma_Y} = -\frac{T - \mu_Y}{\sigma_Y} \,,$$

and the property of Φ that $\Phi(-x) = 1 - \Phi(x)$.

Two points are worth mentioning about Eq. 12.

- Although our derivation and Fig. 1a is assuming $\mu_X < \mu_Y$, it is easy to derive that when $\mu_X \geq \mu_Y$, Eq. 12 still holds. And, it always has the same sign as $\mu_Y - \mu_X$. Hence, Eq. 12 computes d_{DTV} instead of d_{TV}.
- In practice we use the error function, defined as

  $$\mathrm{erf}(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^{x} e^{-t^2} \, \mathrm{d}t \,, \tag{13}$$

  and it satisfies that

  $$\Phi(x) = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right) \,. \tag{14}$$

  Thus, we have

  $$d_{DTV}(p_X, p_Y) = \mathrm{erf}\left(\frac{\mu_Y - \mu_X}{\sqrt{2}(\sigma_X + \sigma_Y)}\right) \,. \tag{15}$$

  The error function erf is built-in and efficient in most major programming languages, which facilitates the calculation of d_{DTV} using Eq. 15.

[3]More details can be found in the supplementary material.

[4]Detailed derivations are in the supplementary material.

Algorithm 1 Visual representation using D3

```
1: Input: An image or video $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots\}$; and,
   a dictionary (visual code book) with size $K$ and cluster
   mean $\boldsymbol{\mu}_k$ and standard deviations $\boldsymbol{\sigma}_k$ $(1 \le k \le K)$
2: for $i = 1, 2, \ldots, K$ do
3:     $X' = \{\boldsymbol{x}_j | \boldsymbol{x}_j \in X, \arg\min_{1 \le k \le K} \|\boldsymbol{x}_j - \boldsymbol{\mu}_k\| = i\}$
4:     Compute the mean and standard deviation vectors of
       the set $X'$, denote as $\boldsymbol{\mu}'$ and $\boldsymbol{\sigma}'$, respectively
5:     $\boldsymbol{f}_i = \operatorname{erf}\left(\frac{\boldsymbol{\mu}' - \boldsymbol{\mu}_i}{\sqrt{2(\boldsymbol{\sigma}' + \boldsymbol{\sigma}_i)}}\right)$, $\boldsymbol{f}_i = \frac{\boldsymbol{f}_i}{\|\boldsymbol{f}_i\|}$
       Note that the operation of erf function and the vec-
       tor division is applied to every component of the
       $\frac{\boldsymbol{\mu}' - \boldsymbol{\mu}_i}{\sqrt{2(\boldsymbol{\sigma}' + \boldsymbol{\sigma}_i)}}$ vector(s) individually
6: end for
7: $\boldsymbol{f} = [\boldsymbol{f}_1^T \ \boldsymbol{f}_2^T \ \ldots \ \boldsymbol{f}_K^T]$, $\boldsymbol{f} = \frac{\boldsymbol{f}}{\|\boldsymbol{f}\|}$
8: Output: The new representation $\boldsymbol{f} \in \mathbb{R}^{d \times K}$
```

We also want to note there has been research to model the discriminative distance between two sets of instance vectors. In (Póczos et al. 2012), non-parametric kernels are estimated from two sets, and use the Hellinger's distance or the Rényi-α divergence to measure the distance between two distributions. This method, however, suffers from one major limitation. Non-parametric kernel estimation is very time consuming, which took 3.3 days in a subset of the Scene 15 dataset, a fact that renders it impractical for large problems. As a direct comparison, D3 only requires less than 2 minutes.

The pipeline using d_{DTV} for visual recognition

The pipeline using d_{DTV} to generate image or video representation follows three steps.

- **Dictionary generation.** We collect a large set of instance vectors from the training set, and then use the k-means algorithm to generate a dictionary that partitions the space of instance vectors into K regions. We compute the mean and standard deviation of the instance vectors inside cluster k as $\boldsymbol{\mu}_k$ and $\boldsymbol{\sigma}_k$ for all $1 \le k \le K$. Values in the standard deviation vector $\boldsymbol{\sigma}_k$ is computed for every dimension independently.
- **Visual representation.** Given any image or video X, we use Algorithm 1 to convert it to a vector representation. In line 5 of this algorithm, the vectors $\boldsymbol{f}_i$ are normalized because k-means is not balanced. Some code word will have a lot of vectors while some will have few. The normalization removes the adverse effect caused by this imbalance.
- **Recognition.** We use the linear SVM classifier.

In Algorithm 1, we use the k-means algorithm to generate a visual codebook, and an instance vector is hard-assigned to one visual code word. A GMM model can also be used as a soft codebook, similar to what is performed in FV. However, a GMM has higher costs in generating both the dictionary and visual representation.

Efficiency and hybrid representation

Since the error function implementation is efficient, the computational cost of D3 is roughly the same as that of VLAD, which is much more efficient than the FV method. The evaluation in (Peng et al. 2014) showed that the time for VLAD is only less than 5% of that of FV. Thus, a visual representation using D3 is efficient to compute.

It is also worth noting that although D3 and FV both used first- and second-order statistics, they use them in very different way (discriminative vs. generative). However, the similarity between D3 and VLAD is much higher than that between D3 and FV. The k-means method is used in both D3 and VLAD.

By computing the D3 and FV representation separately and then concatenate them together to form a hybrid one, we utilize both discriminative and generative information, and can get higher recognition accuracy than D3 and FV, as will be shown in our experiments. D3 is robust (*i.e.*, achieving high accuracy in different types of instance vectors). The combination of D3 and FV is also robust, unlike FV or VLAD.

Experimental Results

To compare the representations fairly, we compare them using the same number of dimensions. For example, the following setups will be compared to each other.

- D3 (or VLAD) with $K_1 = 256$ visual words; the representation has dK_1 dimensions;
- FV with $K_2 = 128$ components ($2dK_2 = dK_1$);
- A mixture of D3 and FV with $K_3 = 128$ in D3 and $K_4 = 64$ in FV ($dK_3 + 2dK_4 = 2dK_2 = dK_1$).

We will use D3's K size to indicate the size of all the above setups (*i.e.*, $K = 256$ in this example).

Two sets of experiments using different types of instance vectors are presented. D3 is evaluated in action recognition (with the ITF instance vectors) and in image recognition (with CNN features as instance vectors). Discussions are provided in the end.

Action recognition

For action recognition, improved trajectory features (ITF) with default parameters (Wang and Schmid 2013) are extracted for one video and then converted to D3, VLAD, FV, and two hybrid representations (D3+FV and VLAD+FV).

We experimented on three datasets: UCF 101 (Soomro, Zamir, and Shah 2012), HMDB 51 (Kuehne et al. 2011) and Youtube (Liu, Luo, and Shah 2009). For UCF 101, the three splits of train and test videos in (Jiang et al. 2013) are used and we report the average accuracy. This dataset has 13320 videos and 101 action categories. The HMDB 51 dataset has 51 actions in 6766 clips. We use the original videos and follow (Kuehne et al. 2011) to report average accuracy of its 3 predefined splits of training / testing videos. Youtube is a small scale dataset with 11 action types. There are 25 groups in each action category and 4 videos are used in each group. Following the original protocol, we report the average of the

25-fold leave one group cross validation accuracy rates. Results on these datasets are reported in Table 1. We summarize the experimental results into the following observations.

D3 is better than VLAD in almost all cases. In the 12 comparisons between D3 and VLAD, D3 wins in 11 cases. D3 often has a margin even if it uses half of number of dimensions of VLAD (*e.g.*, D3 $K = 128$ vs. VLAD $K = 256$). It shows that the D3 representation is effective in capturing useful information for classification.

D3's accuracy is close to that of FV. When the ITF instance vectors are evaluated in action recognition, FV has higher accuracies than that of VLAD, usually 2–3% higher, as shown in Table 1. On average, D3 is only 1% worse than FV. On the Youtube dataset D3 is better than FV (91.55% vs. 91.00%). In terms of computational efficiency, in practice there is no noticeable difference between the running time of D3 and VLAD, and both are much faster than FV. Although the accuracy of D3 is slightly lower than that of FV, we will see that when FV performs poorly with the CNN instance vectors, D3 still achieves high accuracy. That is, D3 is robust and efficient.

The hybrid D3 / FV representation (nearly) consistently outperforms all other methods. Furthermore, we show that the hybrid D3+FV method is the best performer in Table 1. The D3+FV representation is very effective: it is the winner in 8 out of 9 cases. With $K = 128$ in the Youtube set being the only exception, D3+FV consistently beats other methods, including FV and VLAD+FV.

Three points are worth pointing out. First, the success of D3+FV shows that the discriminative (D3) and generative (FV) information are complementary to each other. Since the running time of D3+FV is only roughly half of that of FV in practice, D3+FV is attractive in both speed and accuracy. Second, VLAD+FV is obviously inferior to D3+FV. Its accuracy is almost identical to that of FV. This may suggest that combining only generative information is not very effective.

Image recognition

Now we test how D3 (and the comparison methods) work with instance vectors that are extracted by state-of-the-art deep learning methods. We use the DSP (deep spatial pyramid) method (Gao et al. 2015), which spatially integrates deep fully convolutional networks. A set of instance vectors are efficiently extracted, each of which corresponds to a spatial region (*i.e.*, receptive field) in the original image. The CNN model we use is imagenet-vgg-verydeep-16 in (Simonyan and Zisserman 2015) till the last convolutional layer, and the input image is resized such that its shortest edge is not smaller than 314 pixels, and its longest edge is not larger than 1120 pixels. Six spatial regions are used, corresponding to the level 1 and 0 regions in (Wu and Rehg 2011). (Gao et al. 2015) finds that FV/VLAD usually achieves optimal performance with very small K sizes in DSP. Hence, we test $K \in \{4, 8\}$. The following image datasets are used:

- Scene 15 (Lazebnik, Schmid, and Ponce 2006). It contains 15 categories of scene images. We use 100 training images per category, the rest are for testing.
- MIT indoor 67 (Quattoni and Torralba 2009). It has 15620 images in 67 indoor scene types. We use the train/test split provided in (Quattoni and Torralba 2009).
- Caltech 101 (Fei-Fei, Fergus, and Perona 2004). It consists of 9K images in 101 object plus a background category. We train on 30 and test on 50 images per category.
- Caltech 256 (Griffin, Holub, and Perona 2007). It is a superset of Caltech 101, with 31K images, and 256 object plus 1 background categories. We train on 60 images per category, the rest for testing.
- SUN 397 (Xiao et al. 2010). It is a large scale scene recognition dataset, with 397 categories and at least 100 images per category. We use the first 3 train/test splits of (Xiao et al. 2010).

Except for the indoor and SUN datasets, we run 3 random train/test splits in each dataset. Average accuracy rates on these datasets are reported in Table 2. As shown by the standard deviation numbers in Table 2, the deep learning instance vectors are stable and the standard deviations are small in most cases. Thus, we tested with 3 random train/test splits instead of more (*e.g.*, 5 or 10). The same imagenet-vgg-verydeep-16 deep network was used as the CNN baseline.

D3 and D3+FV have shown excellent results when combining with instance vectors extracted by deep nets. We have the following key observations from Table 2, which mostly coincide well with the observations concerning action recognition in Table 1. The last row in Table 2 shows the current state-of-the-art recognition accuracy in the literature, which are achieved by various systems that depend on deep learning using the same evaluation protocol.

D3 is slightly better than FV. D3 is better than FV in 3 datasets (Scene 15, indoor 67 and SUN 397), but worse than FV in the two Caltech datasets. It is worth noting that D3's accuracy is higher than that of FV by a larger margin in indoor 67 (1–2%) and SUN 397 (1.5–2.2%), while FV is only higher than D3 by 0.3–0.7% in the Caltech 101 and 256 datasets. Another important observation is that the win/loss are consistent among the train/test splits. In other words, if D3 wins (loses) in one dataset, it wins (loses) consistently in all three splits.[5] Thus, the CNN instance vectors lead to stable comparison results, and we believe 3 train/test splits are enough to compare these algorithms.

VLAD is better than both D3 and FV, but D3 is closer to VLAD than FV. Although FV outperforms VLAD in Table 1, a reversed trend is shown in Table 2 using CNN instance vectors. That is, FV and VLAD are not very robust. VLAD is almost consistently better than FV, up to 3.2% higher in the SUN 397 dataset. The accuracy of D3, however, is much closer to that of VLAD than that of FV. D3 is usually 0.3%–0.6% lower than VLAD, with only two cases up to 1.1% ($K = 8$ in Caltech 256 and SUN 397). In short, we find that *D3 is indeed robust*, fulfilling its designing objective.

The D3+FV hybrid method is the overall recommended method again. The second part of Table 2 presents results of hybrid methods. D3+FV has the highest accuracy in 6 cases, while VLAD+FV has only one winning case. When

[5]Detailed per-split accuracy numbers are omitted.

Table 1: Action recognition accuracy (%) comparisons. Note that the results in one column are compared with the same number of dimensions in the representations. For example, the column K=256 means that $K = 256$ for D3 and VLAD, $K = 128$ for FV, and in the hybrid representation, $K = 128$ for D3 or VLAD combined with $K = 64$ for FV. The best results are shown in bold face.

	UCF 101				HMDB 51				Youtube			
K	512	256	128	64	512	256	128	64	512	256	128	64
D3	84.35	84.32	83.03	81.34	56.14	55.29	54.71	51.70	89.91	91.55	**91.09**	90.36
VLAD	82.81	82.54	81.59	79.78	55.45	55.14	53.92	50.22	90.00	89.73	89.18	89.09
FV	85.23	84.82	83.80	82.48	58.13	57.34	55.88	53.20	91.00	91.00	90.73	90.45
D3+FV	**85.92**	**85.44**	**84.20**		**58.34**	**57.63**	**56.58**		**91.73**	**91.36**	90.45	
VLAD+FV	85.23	84.54	83.52		58.13	57.60	55.64		90.91	**91.36**	90.82	

Table 2: Image recognition accuracy (percent) comparisons. The definition of K is the same as that used in Table 1. The best results are shown in bold face. Standard deviations are also showed after the ± sign.

	Scene 15		MIT indoor 67		Caltech 101		Caltech 256		SUN 397	
	$K = 4$	$K = 8$	$K = 4$	$K = 8$	$K = 4$	$K = 8$	$K = 4$	$K = 8$	$K = 4$	$K = 8$
D3	92.34±0.23	92.10±0.65	77.31	77.76	93.60±0.17	93.80±0.58	83.15±0.15	82.92±0.09	59.93±0.24	60.22±0.07
VLAD	92.58±0.60	92.61±0.42	**77.61**	**78.13**	94.20±0.39	94.11±0.57	84.01±0.02	84.00±0.10	60.61±0.25	61.22±0.33
FV	91.96±0.40	91.53±0.56	75.97	75.82	94.32±0.51	94.10±0.33	83.75±0.16	83.40±0.13	58.40±0.12	57.97±0.28
D3+FV	**92.83**±0.55	**92.82**±0.31	77.09	77.99	**94.72**±0.51	**94.51**±0.44	**84.77**±0.12	**84.62**±0.15	**61.48**±0.22	61.38±0.52
VLAD+FV	92.82±0.52	92.76±0.56	77.54	78.06	94.71±0.41	94.45±0.51	84.18±0.51	84.61±0.16	61.32±0.26	**61.83**±0.27
CNN baseline	89.88±0.76		69.78		90.55±0.31		82.02±0.12		53.90±0.45	
State-of-the-art	91.59 (Zhou et al. 2014)		77.56 (Gong et al. 2014)		93.42 (He et al. 2014)		77.61 (Chatfield et al. 2014)		53.86 (Zhou et al. 2014)	

comparing D3+FV with D3, FV or VLAD in detail, this hybrid method has higher accuracy than any single method in all train/test splits in all 36 comparisons (4 datasets excluding the indoor 67 dataset × 3 individual representations × 3 train/test splits). The MIT indoor 67 dataset is a special case, where VLAD is better than all other methods. We are not yet clear what characteristic of this dataset makes it particularly suitable for VLAD.

The fact that D3 is in general inferior to VLAD in this setup also indicates that CNN instance vectors have different characteristics than the ITF instance vectors in videos, for which VLAD is inferior to D3.

This might be caused by the fact that D3 and VLAD used very small K values ($K = 4$ or 8) with CNN instance vectors, compared to $K \geq 64$ in Table 1. Hence, both methods have much fewer number of dimensions now, and a few VLAD dimensions with highest discriminative powers may lead to better performance than D3. We will leave a careful, more detailed analysis of this observation to future work.

Significantly higher accuracy than state-of-the-art, especially in those difficult datasets. DSP (Gao et al. 2015) (with D3 or other individual representation methods) is a strong baseline, which already outperforms previous state-of-the-art in the literature (shown in the last row of Table 2). The hybrid method D3+FV leads to even better performance, *e.g.*, its accuracy is 7.6% higher than the place deep model of (Zhou et al. 2014) for SUN 397.

Discussions

Overall, the proposed D3 representation method has the following properties:

- **D3 is discriminative, efficient, and *robust*.** D3 is not the individual representation method that leads to the highest accuracy. FV is the best in our action recognition experiments with ITF instance vectors, while VLAD is the best in our image categorization experiments using CNN features. It is, however, *the most robust one*. It is only slightly worse than FV in action recognition and slightly worse than VLAD in image categorization. Although VLAD is outperformed by FV by a large margin in action recognition (Table 1) and vice versa for image categorization (Table 2), D3 has stably achieved high accuracy rates in both sets of experiments. Since we do not know *a priori* whether FV or VLAD is suitable for a specific problem at hand, a robust representation such as D3 is useful. D3 is also as efficient as VLAD, and is much faster than the FV method;
- **D3+FV is the overall recommended method.** Using the same number of dimensions for all individual and hybrid methods, D3+FV has shown the best performance, which indicates that the information encoded by D3 (discriminative) and FV (generative) forms a synergy. Since the FV part of D3 only uses half the number of Gaussian components than that in individual FV, D3+FV is still more efficient than FV alone.

In short, D3 and D3+FV are robust, effective and efficient in encoding entities that are represented as sets of instance vectors.

Conclusions and Future Work

We proposed the Discriminative Distribution Distance (D3) method to encode an entity (which comprises of a set of instance vectors) into a vector representation. Unlike existing methods such as FV or VLAD that are designed from a generative perspective, D3 is based on discriminative ideas to make the representation robust (*i.e.*, always achieving high accuracy) when different types of instance vectors are used. We proposed a new directional distance to measure how two distributions (sets of vectors) are different with each other, and proposed to use the MPM classifier to robustly estimate this distance, even in worst-case scenarios.

These design choices lead to excellent classification accuracy of the proposed D3 representation, which are verified by extensive experiments on action and image categorization datasets. D3 is also efficient, and the hybrid D3+FV representation has achieved the best results among compared individual and hybrid methods.

In the same spirit as D3, we plan to combine D3 and FV in a more principled way, which will add discriminative perspectives to FV, to make it more robust, and to further reduce the computational cost of the hybrid representation using D3+FV. Currently the hybrid representation D3+VLAD is inferior to D3+FV. We conjecture this may be caused by the similarity between D3 and VLAD. We will further study how the benefits of VLAD can be utilized (*e.g.*, when CNN instance vectors are used).

References

Chatfield, K.; Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2014. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*.

Cimpoi, M.; Maji, S.; and Vedaldi, A. 2015. Deep convolutional filter banks for texture recognition and segmentation. In *CVPR*.

Csurka, G.; Dance, C. R.; Fan, L.; Willamowski, J.; and Bray, C. 2004. Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*.

DasGupta, A. 2011. *Probability for statistics and machine learning: fundamentals and advanced topics*. Springer Science & Business Media.

Fei-Fei, L.; Fergus, R.; and Perona, P. 2004. Learning generative visual models from few training example: an incremental Bayesian approach tested on 101 object categories. In *CVPR 2004, Workshop on Generative-Model Based Vision*.

Gao, B.-B.; Wei, X.-S.; Wu, J.; and Lin, W. 2015. Deep spatial pyramid: The devil is once again in the details. *arXiv:1504.05277*.

Gkioxari, G., and Malik, J. 2015. Finding action tubes. In *CVPR*.

Gong, Y.; Wang, L.; Guo, R.; and Lazebnik, S. 2014. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, LNCS 8695, 392–407.

Griffin, G.; Holub, A.; and Perona, P. 2007. Caltech-256 object category dataset. Technical Report CNS-TR-2007-001, Caltech.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2014. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, volume LNCS 8691, 346–361.

Jégou, H.; Perronnin, F.; Douze, M.; Sánchez, J.; Pérez, P.; and Schmid, C. 2012. Aggregating local images descriptors into compact codes. *IEEE TPAMI* 34(9):1704–1716.

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; and Darrell, T. 2014. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, 675–678.

Jiang, Y.-G.; Liu, J.; Zamir, A. R.; Laptev, I.; Piccardi, M.; Shah, M.; and Sukthankar, R. 2013. THUMOS: The first international workshop on action recognition with a large number of classes.

Kuehne, H.; Jhuang, H.; Garrote, E.; Poggio, T.; and Serre, T. 2011. HMDB: a large video database for human motion recognition. In *ICCV*, 2556–2563.

Lanckriet, G.; Ghaoui, L. E.; Bhattacharyya, C.; and Jordan, M. 2002. A robust minimax approach to classification. *Journal of Machine Learning Research* 3:555–582.

Lazebnik, S.; Schmid, C.; and Ponce, J. 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, volume II, 2169–2178.

Liu, J.; Luo, J.; and Shah, M. 2009. Recognizing realistic actions from videos "in the wild". In *CVPR*, 1996–2003.

Lowe, D. G. 2004. Distinctive image features from scale-invariant keypoints. *IJCV* 60(2):91–110.

Peng, X.; Wang, L.; Qiao, Y.; and Peng, Q. 2014. Boosting VLAD with supervised dictionary learning and high-order statistics. In *ECCV*, LNCS 8691, 660–674.

Póczos, B.; Xiong, L.; Sutherland, D. J.; and Schneide, J. 2012. Nonparametric kernel estimators for image classification. In *CVPR*, 2989–2996.

Quattoni, A., and Torralba, A. 2009. Recognizing indoor scenes. In *CVPR*, 413–420.

Sánchez, J.; Perronnin, F.; Mensink, T.; and Verbeek, J. 2013. Image classification with the fisher vector: Theory and practice. *IJCV* 105(3):222–245.

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

Soomro, K.; Zamir, A. R.; and Shah, M. 2012. UCF101: A dataset of 101 human action classes from videos in the wild. Technical report, CRCV-TR-12-01, University of Central Florida.

Wang, H., and Schmid, C. 2013. Action recognition with improved trajectories. In *ICCV*, 3551–3558.

Wu, J., and Rehg, J. M. 2011. CENTRIST: A visual descriptor for scene categorization. *IEEE TPAMI* 33(8):1489–1501.

Xiao, J.; Hays, J.; Ehinger, K. A.; Oliva, A.; and Torralba, A. 2010. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 3485–3492.

Xu, Z.; Yang, Y.; and Hauptmann, A. G. 2015. A discriminative CNN video representation for event detection. In *CVPR*.

Zhou, B.; Lapedriza, A.; Xiao, J.; Torralba, A.; and Oliva, A. 2014. Learning deep features for scene recognition using Places database. In *NIPS 27*, 487–495.