Towards Efficient Learning of Optimal Spatial Bag-of-Words Representations

Lu Jiang[1], Wei Tong[1], Deyu Meng[2], Alexander G. Hauptmann[1]
[1] School of Computer Science, Carnegie Mellon University
[2] School of Mathematics and Statistics, Xi'an Jiaotong University
{lujiang, weitong, alex}@cs.cmu.edu, dymeng@mail.xjtu.edu.cn

ABSTRACT

Spatial Pyramid Matching (SPM) assumes that the spatial Bag-of-Words (BoW) representation is independent of data. However, evidence has shown that the assumption usually leads to a suboptimal representation. In this paper, we propose a novel method called Jensen-Shannon (JS) Tiling to learn the BoW representation from data directly at the BoW level. The proposed JS Tiling is especially appropriate for large-scale datasets as it is orders of magnitude faster than existing methods, but with comparable or even better classification precision. Experimental results on four benchmarks including two TRECVID12 datasets validate that JS Tiling outperforms the SPM and the state-of-the-art methods. The runtime comparison demonstrates that selecting BoW representations by JS Tiling is more than 1,000 times faster than running classifiers. Besides, JS Tiling is an important component contributing to CMU Teams' final submission in TRECVID 2012 Multimedia Event Detection.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing- *Indexing methods*; I.2.5 [**Artificial Intelligence**]: Learning- *Feature Representation learning*

General Terms

Algorithms, Measurement, Experimentation

Keywords

Bag of Visual Words, Feature Representation, SPM, Spatial Pyramid, Jensen-Shannon Tiling, Pooling Method

1. INTRODUCTION

The Spatial Bag-of-Words (BoW) model has proven one of the most broadly used models in image and video retrieval. In this model, an image is geometrically partitioned into several tiles (or grids) described by histograms of visual words. The whole image is then represented by the concatenated histograms from all the tiles. A further extension rendering the approach more robust is the Spatial Pyramid Matching (SPM) [12]. The SPM assumes that the spatial BoW representation is independent of data. However, evidence has shown that manually defined representations [24, 21, 7, 30, 13] considering salient spatial layouts outperform the predefined BoW representation on many problems.

A straightforward solution is to select optimal BoW representations in terms of their classification precision on the validation set. This strategy, however, is computationally infeasible in practice due to the large search space. Recently, several approaches have been proposed in order to learn the representation in a more tractable way [18, 10]. The idea is to optimize the BoW representation jointly with the subsequent classifier. However, learning the representation on large-scale datasets is still computationally challenging [10]. Moreover, certain greedy methods have to be employed to approximate the solution when the dataset becomes larger [10] as the time-consuming classifier training, or equivalently, quadratic programming problem solving, is a part of the learning process. For example, learning the best representation for a large-scale TRECVID video dataset named Multimedia Event Detection generally takes several weeks on a single desktop [21]. Therefore, although the existing methods can yield some improvement over the SPM, the expensive overhead of learning restricts its applicability in large-scale problems.

In this paper, we propose a novel method called *Jensen-Shannon (JS) Tiling* to learn the spatial BoW representation from data directly at the BoW level. The proposed method is well chosen for large-scale datasets as it is orders of magnitude faster than the existing methods, but with comparable or even better performance. Specifically, we call the way to partition an image/video a tiling which corresponds to a spatial BoW representation. For example, 2×2 and 4×4 grids in the SPM are two examples of tilings. As shown in Fig. 1, JS Tiling takes the visual words extracted from the raw images/videos as the input and outputs the learned tiling. JS Tiling consists of two steps. First it systematically generates all possible tilings from a base mask. For example, all tilings in Fig. 1 are example tilings derived from 4×4 grids by combining the tiles in it. Second, the generated tilings are evaluated by a proxy based on the JS divergence, which estimates the divergence between the average positive and negative word distribution generated by the tiling. Note, as in existing methods, the feature extraction and coding need to be conducted only once. The over-

ICMR '14, April 01-04, 2014, Glasgow, United Kingdom
Copyright 2014 ACM 978-1-4503-2782-4/14/04 ...$15.00
http://dx.doi.org/10.1145/2578726.2578739.

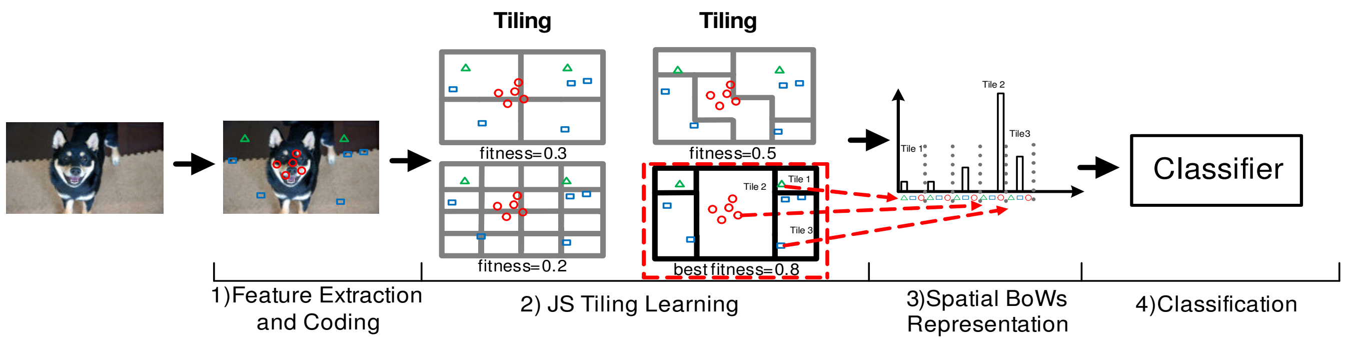


Figure 1: The pipeline of image/video classification with the proposed JS Tiling. Each rectangle corresponds to a tiling and the number below denotes the fitness score.

head of learning representations depends on the complexity of the learning algorithm.

Compared with existing methods [8, 10, 18], the proposed JS Tiling is orders of magnitude faster because the fitness of the BoW representation is directly captured at lower BoW level, independent of the subsequent classifier. The complexity of evaluating a tiling on the pre-computed histograms is independent of the size of dataset rendering it appropriate for large-scale datasets. Theoretically we prove that the negative JS divergence of BoW distributions is approximately the upper bound of the training error of a weighted K-Nearest Neighbor (K-NN) classifier. Therefore, while minimizing the negative JS divergence, the training error of a weighted K-NN classifier tends to be also minimized. The result is important because it connects a divergence at the lower level of BoW to the training error of a weighted K-NN classifier at higher recognition level, and provides a justification why the computationally inexpensive divergence can be a proxy to the computationally expensive classifier. In addition, as learning the representation in JS Tiling is independent of learning the subsequent classifier, the learned spatial BoW representation can be expected to be discriminative in general and not tuned to a specific type of predictive model.

Experimental results demonstrate that selecting the BoW representation by the proposed JS Tiling is more than 1,000 times faster than running classifiers. JS Tiling consistently outperforms the SPM on four diverse datasets on scene/object recognition and event detection. With the help of the learned BoW representation, we are able to achieve state-of-the-art performance across datasets, including two challenging TRECVID datasets. Besides, JS Tiling is an important component contributing to CMU Teams' final submission in TRECVID 2012 Multimedia Event Detection [21, 29]. In summary, the contribution of this paper is twofold:

- We formulate spatial tiling generation as a set partition problem and provide an algorithm to generate all possible tilings from a given mask.
- This paper is a first attempt to evaluate the spatial BoW representation directly at the lower BoW level. Theoretically we justify the JS divergence by proving that the negative JS divergence of BoW distributions is approximately the upper bound of the training error of a weighted K-Nearest Neighbor classifier.

2. RELATED WORK

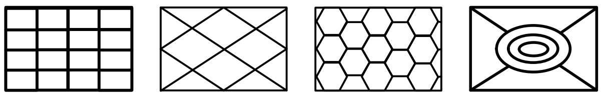
Evidence has shown that the manually designed representations generally outperform predefined BoW representations in many problems. For example, Viitaniemi et al. observed that manually designed tilings achieve reasonable improvement over the SPM on the Pascal VOC dataset [24]. Similar observations have been confirmed on other datasets. Zhang et al. found that the manually designed spatial BoWs representation is more effective than the SPM for surveillance event detection [30]. Tong et al. confirmed the same observation on the multimedia event detection task [21].

Recently, several approaches have been proposed to automatically learn an optimal representation from data. The idea is to find a function or operator that produces informative statistics in a specific spatial area. For example, Feng et al. proposed a geometric l_p-norm pooling that finds the operator by preserving the geometric information [8]. Jia et al. focused on finding the spatial regions called receptive fields [10]. The method can be regarded as a embedded feature selection method which learns optimal receptive fields jointly with the subsequent classifier. Sharma et al. proposed to learn the region on a coarse level [18], where the image is recursively split into the finer grids, and the problem is modeled using quadratic programming and solved jointly with the SVM classifier. As training a classifier is a part of learning, these methods are orders of magnitude slower than the proposed method. Therefore, although the above pooling method can yield some improvement over the SPM, the expensive overhead of learning restricts its applicability in many large-scale problems.

3. PROBLEM FORMULATION

One can imagine the problem as an analogy to a bathroom floor tiling problem, where the task is to find different ways to tile the image (floor) with different styles of tiles (mosaics). An arrangement of the tiles is called a *tiling* (or *spatial tiling*) which corresponds to a spatial BoW representation. We define the floor as the mask from which more tilings can be derived by combining the tiles in it. Formally, a mask is defined as a tuple of n tiles $S = (t_1, ..., t_n)$ with each tile t_i covering a non-overlapping area. A mask is a tiling and any tiling can be a mask. Fig. 2 illustrates several example masks. Non-rectangular masks may not be better than rectangular masks and the introduction of non-rectangular masks is only to expand the tiling search space.

Each tile in the mask is associated with an ID indicating



(a) rectangle (b) diamond (c) hexagon (d) ellipse

Figure 2: Examples of the different types of masks.

its geometrical location on the mask, e.g. see the numbers in the tiles in Fig. 3. Tiles in a mask can be combined to form a bigger tile. Following the adjacency constraint in [8], we require that a tile can only be combined from adjacent tiles in the mask. Formally given a mask S, two tiles $t_i, t_j \in S$ are adjacent if they share at least a common edge. A set of tiles U is called an adjacent set if it either contains a single tile or $\exists t_i, t_j \in U (t_j \neq t_i)$ such that t_i and t_j are adjacent and $(U \backslash t_i)$ is an adjacent set. For example, in Fig. 3(c) $\{t_3, t_5, t_6\}$ is an adjacent set whereas $\{t_1, t_2, t_9\}$ is not. From the graph perspective, adjacent tile sets are identical to the connected components in the graph where the node is the tile and the edge indicates the two tiles are adjacent. If we regard a mask as a set and its tiles as the elements in the set, we can model a tiling as a set partition over the mask with the adjacency constraint. Formally we have:

Definition 1 (Tiling). *A tiling is a mapping $\mathcal{T}_\kappa(S) = S'$, denoted by $\kappa = \langle \kappa_1, ..., \kappa_{|S|} \rangle$, from a source mask $S = (t_i)$ to a target mask $S' = (t'_j)$, such that $t'_j = \{t_i | \kappa_i = (j-1), \kappa_i \in \mathbb{Z}\}$. $\mathbb{Z}$ is the set of integers. A valid tiling $\mathcal{T}_\kappa$ satisfies:*

1. *$\kappa = \{i \in [2, n], \kappa_i \in \mathbb{Z}, \kappa_1 \leq \kappa_i \leq \max_{1 \leq j < i} \kappa_j + 1\}$;*
2. *$\forall j \in [1, |S'|]$, t'_j is an adjacent set.*

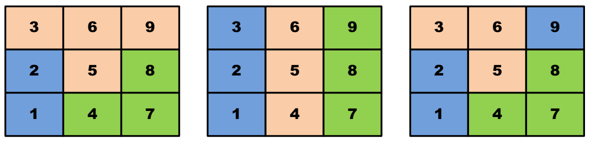
The first condition in Definition 1 is inherited from the set partition theory [16, 20], which requires the target mask to be fully covered by non-overlapping tiles. The second condition indicates that every tile in the target mask should be adjacent. Given a source mask, a tiling is represented by a vector κ, also known as codewords in [6], each dimension indicating the tile's membership in the target mask.

Fig. 3 illustrates some examples of tilings derived from a 3×3 rectangle mask, where each numbered region t_i is a tile in the source mask and the regions with the same color constitute a tile in the target mask t'_i. For example, the carnation tile in Fig. 3(a) consists of four tiles 3,5,6 and 9 (i.e. $t'_2 = \{t_3, t_5, t_6, t_9\}$) as in the tiling $\kappa = \langle$ 0,0,1,2,1,1,2,2,1$\rangle$ the third, fifth, sixth and ninth dimension share the same membership 1. Fig. 3(c) is not a valid tiling as $t' = \{t_1, t_2, t_9\}$ is not an adjacent set. Particularly if every tile in the target mask comprises of the same number of tiles, i.e. $\forall t_i, t_j \in S', |t_i| = |t_j|$, we call $\mathcal{T}_\kappa$ an *equal tiling*. For example, Fig. 3(b) is an equal tiling and most of the tilings used in the literature are equal tilings [12, 21]. The word tiling refers to the tiling operator $\mathcal{T}_\kappa$ in Definition 1, and, without causing ambiguity, given a source mask S it also refers to the spatial representation in the target mask $\mathcal{T}_\kappa(S)$.

Based on the above definition, we can summarize the spatial tiling learning problem as: given a mask S, find an optimal tiling $\mathcal{T}_\kappa{}^*$ such that

$$\mathcal{T}_\kappa^* = \arg\min_{\tilde{\mathcal{T}}_\kappa} cost(\tilde{\mathcal{T}}_\kappa), \tag{1}$$

where the cost function is calculated from a certain evaluation criterion as introduced in the next section.



(a) ⟨0,0,1,2,1,1,2,2,1⟩ (b) ⟨0,0,0,1,1,1,2,2,2⟩ (c) Not a tiling.

Figure 3: Examples of valid and invalid tilings derived from a 3 × 3 rectangle mask. The tiles in the mask are numbered by their IDs and the color indicates the membership for each tile after tiling.

4. SPATIAL TILING

To tackle the above problem, we discuss solutions to the following sub-problems: how to explore the tiling search space and given a tiling how to efficiently evaluate its cost.

4.1 Tiling Function Domain

Generally, the spatial tiling learning problem is a NP-Hard problem and identical to the set partition problem [20] if the adjacency criterion in Definition 1 is ignored. However, the problem is still tractable given the reasonable masks including the commonly used masks in the literature [12, 24, 30, 21]. The numbers of all possible set partitions, tilings and equal tilings on different masks are listed in Table 1, where the Parameter column lists the parameters used in generating the masks[1], e.g. Rectangle 4×4 corresponds the mask in Fig. 2(a). As we see, the adjacency constraint significantly reduces the search space. For example, a 4×4 rectangle mask has around 1.6 million tilings but only 255 equal tilings, in contrast with 10.4 billion set partitions.

Type	Parameter	#Set Partition	#Tiling	#Equal Tiling
Rectangle	2×2	15	12	4
Rectangle	3×3	21147	1434	12
Rectangle	4×4	10480142147	1691690	225
Diamond	1×1	15	12	4
Diamond	2×2	52	16	2
Diamond	3×3	4213597	17326	23
Hexagon	1	52	20	2
Hexagon	1.5	4140	466	7
Ellipse	4	4140	344	5
Ellipse	8	4213597	5504	10

Table 1: The number of all possible set partitions, tilings and equal tilings on different types of masks. All tilings are available at [1].

To explore the search space, inspired by the algorithm proposed in [6, 16], we introduce Algorithm 1 to generate all tilings in Table 1. Note that given a mask, the tilings only needs to be generated once and the pre-generated tilings can be reused on different datasets. In Algorithm 1, a tiling is represented by a vector κ and $\mathbf{m}$ is an auxiliary vector. Having initialized κ and the auxiliary vector, in Step 5, the algorithm invokes the function nextPartition to find the next set partition and checks whether κ obeys the adjacent criterion via Step 6 to Step 8. Note that the adjacent criterion checker in Step 6 can be implemented as a simple recursive connected component traverse subroutine. The source code of Algorithm 1 and the pre-generated tilings in Table 1 are available at [1].

[1]Detailed information about the mask parameters is at [1]

Algorithm 1: Tiling generation algorithm

```
input  : A mask $S$ and let $n = |S|$.
output: All possible tilings $\mathcal{T}_\kappa$ in the result set.
for $i = 1$ to $n$ do
    $m_i = \kappa_i = 0$; // Initialization
end
do
    $\kappa$ = nextPartition($\kappa$, m);
    if $\mathcal{T}_\kappa(S)$ are adjacent sets then
        Add $\mathcal{T}_\kappa$ to the result set;
    end
while $\kappa \neq$ null;

// Start the subroutine
nextPartition ← function($\kappa$, m)
for $i = n$ to 2 do
    if $\kappa_i \leq m_{i-1}$ then
        $\kappa_i \leftarrow \kappa_i + 1$; $m_i \leftarrow \max(\kappa_i, m_i)$;
        if $i + 1 \leq n$ then
            for $j = i + 1$ to $n$ do
                $\kappa_j \leftarrow \kappa_1$; $m_j \leftarrow m_i$;
            end
        end
        return $\kappa$;
    end
end
return null;
```

4.2 Cost Function

Consider a binary classification scenario, where $\mathbf{x}_i$ represents the i^{th} sample in the training set with the label $y_i \in \{-1, +1\}$. In the spatial BoW representation, a sample is a concatenation of histograms of all tiles in the mask. In each tile, the histogram is represented by the visual words geographically located in the area, denoted by $\mathbf{x}_i^j(k)$, where i indexes the sample, j indexes the tile and k indexes the dimension of the histogram. For the j^{th} tile in the mask, define its positive and negative histogram as the average histogram of all positive and negative samples, respectively,

$$\mathbf{H}_+^j = \frac{1}{n^+} \sum_{y_i=+1} (\mathbf{x}_i^j + \alpha), \tag{2}$$

where n^+ denotes the total number of positive samples. While counting the word frequency, we add the Laplace smoother with $\alpha = 1$ to avoid the underflow. Define $\mathbf{D}_+^j$ as the corresponding (L_1 normalized) word distribution of the histogram $\mathbf{H}_+^j$, i.e. $\mathbf{D}_+^j = \mathbf{H}_+^j / \; \|\mathbf{H}_+^j\|_1$.

An optimal spatial BoW representation tends to separate the positive and negative samples with the maximum distance [3]. In this paper, the distance is derived from the Kullback-Leibler (KL) divergence between the average word distribution of positive and negative samples:

$$KL(\mathbf{D}_+^j \parallel \mathbf{D}_-^j) = \sum_{k=1}^{|V|} \mathbf{D}_+^j(k) \log \frac{\mathbf{D}_+^j(k)}{\mathbf{D}_-^j(k)}, \tag{3}$$

where $|V|$ represents the size of the dictionary. Since the distance is required to be symmetric, we use the Jensen-Shannon (JS) divergence:

$$JS(\mathbf{D}_+^j \| \mathbf{D}_-^j) = \frac{1}{2} KL(\mathbf{D}_+^j \| \frac{\mathbf{D}_+^j + \mathbf{D}_-^j}{2}) + \frac{1}{2} KL(\mathbf{D}_-^j \| \frac{\mathbf{D}_+^j + \mathbf{D}_-^j}{2}). \tag{4}$$

Based on the JS divergence we propose the following cost function for a given mask S:

$$cost(\mathcal{T}_\kappa) = \lambda |\mathcal{T}_\kappa(S)| - \sum_{j=0}^{|\mathcal{T}_\kappa(S)|-1} \frac{JS(\mathbf{D}_+^j \parallel \mathbf{D}_-^j)}{|\mathcal{T}_\kappa(S)|}, \tag{5}$$

where $|\mathcal{T}_\kappa(S)|$ indicates the number of tiles after tiling and λ is a parameter. The second term in Eq. (5) measures the average distance between the positive and the negative word distributions. The first term serves as a regularization term that controls the complexity to avoid overfitting. Therefore, the cost function trades off the model fitness against the model complexity. Given a tiling, the computational complexity of Eq. (5) is $O(NM)$, where N and M denote the number of samples and the dimension of the histogram, respectively. The complexity can be further reduced to $O(M)$ if the histogram of every tile in the mask is pre-calculated. Since the dimension M is usually a small constant, selecting optimal BoW representations by Eq. (5) is orders of magnitude faster than by running classifiers, the complexity of which is generally $O(MN^2)$.

Eq. (5) may not be the optimal proxy for this problem but it offers an efficient and reasonably accurate approach to evaluate BoW representations. Experimental results in Section 5.5 substantiates this argument. JS is a broadly used metric in information retrieval but has not been used to evaluate the spatial BoW representation. Measuring the BoW representation by JS is consistent with the distribution separability principle in [3]. If the histogram is L_1 normalized, we prove that the negative JS divergence in the second term of Eq. (5) is an upper bound of the training error of a weighted K-Nearest Neighbor classifier $K = N$ (see Appendix). Since K is large, the proxy is expected to be robust across datasets. The experimental results substantiate this argument. Our finding is important because it establishes a connection between a divergence working at lower BoW level and the training error of a classifier at higher recognition level, and provides a justification why the computationally inexpensive divergence can be a proxy to the computationally expensive weighted K-NN classifier. In addition, Eq. (5) can be used to evaluate not only tilings but also BoW representations in other problems, e.g. to determine best sliding window size.

If the search space is manageable, such as the search space in Table 1, we can exhaustively search the optimal tiling using Eq. (5). When the exhaustive search becomes computationally infeasible, we can apply gradient descent using the objective function Eq. (5) to find the optimal solution.

5. EXPERIMENTS

In this section, we empirically compare our results with Spatial Pyramid Matching (SPM) and the state-of-the-art method on four datasets ranging from a clean dataset to challenging TRECVID datasets. The datasets are 15-scene categories [12], Surveillance Event Detection (SED) [17, 30], Multimedia Event Detection (MED) [21] and Pascal VOC [7]. Experiments are repeated five times with randomly generated training and test samples. In each run, the samples are partitioned approximately equally into the training and test set. A standard LibSVM classifier [4] with a χ^2 kernel is used and the multi-class classification is conducted using the one-versus-all strategy. The tilings in Table 1 are included as the masks. Since the search space is manageable, we conduct an exhaustive search with Eq. (5). On each dataset, the performance is evaluated by the common metric used on the dataset and the Mean Average Precision (MAP).

5.1 15-Scene Categories

The 15-scene dataset consists of 4,485 scene images from

L	Spatial Pyramid		Rectangle Masks		All Masks	
	Accuracy	MAP	Accuracy	MAP	Accuracy	MAP
0	75.3±0.3	81.5±0.6	80.4±0.7	83.2±0.6	82.4±0.4	85.5±0.4
1	80.7±0.6	83.3±0.6	80.8±0.5	83.6±0.6	82.2±0.5	85.4±0.4
2	**80.8±0.6**	**83.5±0.5**	81.4±0.6	84.1±0.6	82.7±0.6	85.8±0.4
3	80.1±0.6	82.4±0.5	81.5±0.6	84.1±0.7	82.8±0.5	85.8±0.4
4	79.2±0.6	81.2±0.6	81.7±0.6	84.2±0.6	83.5±0.7	86.7±0.5
7	-	-	**81.9±0.5**	**84.6±0.5**	**85.3±0.4**	**88.0±0.3**

Table 3: Comparison of the tiling fusion results on 15-scene. The first column is the level in SPM.

15 categories. Following the setting in [12], we extract dense SIFT with the vocabulary size $|V| = 1024$ and use randomly selected 100 images as the training samples. First we compare the learned tilings with the predefined tilings used in Spatial Pyramid Matching (SPM). Table 2 lists the comparison result. The "Predefined Masks" column shows the performance of the top 5 best predefined tilings. The tilings in the "Rectangle Mask" column are learned on the rectangle masks. The "All Masks" column further includes tilings learned on non-rectangular masks. As we see, the learned tilings consistently outperform the predefined tilings.

Then we compare the fusion of the learned tilings with the SPM at different levels, as shown in Table 3. The first column indicates the level in the SPM. To be consistent, we fuse $L + 1$ learned tilings at level L. The tilings in the "Rectangle Mask" column are learned on the rectangle masks whereas the tilings in the "All Masks" column are learned on all masks. As the tiling fusion is beyond the topic of this paper, we gradually fuse tilings in descending order of their performance on the development set. As we see, the SPM underperforms the fusion of learned tilings on both rectangle and all masks. The result indicates that the BoW representation in the SPM is suboptimal and can be further improved by salient tilings learned from data. A notable observation is that the performance of SPM decreases when L grows larger than 2 due to the lack of remaining salient tilings. However, the performance of learned tilings keeps increasing along with the level L before reaching its saturation point.

Table 4 lists the performance comparison with state-of-the-art methods on the dataset. For a fair comparison, we cite the score with the same vocabulary size 1024 in [2]. We report our best fusion result of the learned tilings in Table 3. As we see, the proposed JS Tiling achieves the highest scores. For example, it improves the accuracy of the SPM by an absolute 4.5%. It also obtains reasonable improvements over the state-of-the-art methods. For example, it improves the MAP of the best baseline [19] by an absolute 2.5%.

5.2 Surveillance Event Detection (SED)

The TRECVID SED [17] dataset consists of 143 hours of surveillance video recorded in London Gatwick Airport captured by five cameras. The task is to detect seven events such as "PersonRuns", "ObjectPut" and "PeopleMeet". Following the setting in [30] we extract the MoSIFT (Motion SIFT) [5] feature on the 2011 development set with the vocabulary size $|V| = 4096$. We randomly sample some non-event examples as the negative samples using a sliding window of 60 frames. The standard metric Minimum DCR (Detection Cost Rate) [17] is used and averaged across 7 events. The smaller Minimum DCR is, the better the method and the perfect method has zero Minimum DCR.

Table 5 presents the comparison between the SPM and the fusion of learned tilings at different levels. We observe

Dataset	Method	MAP	Accuracy
15-Scene	SPM [12]	83.5±0.5	80.8±0.6
	Boureau et al. [2]	-	84.9±0.3
	Sharma et al. [19]	85.5±0.7	-
	van Gemert et al. [23]	-	76.7±0.4
	Sharma et al. [18]	-	81.2±0.6
	Yang et al. [27]	-	80.3±0.9
	JS Tiling	**88.0±0.3**	**85.3±0.4**
	Method	MAP	Min DCR
SED	SPM [12]	22.8±1.0	89.0±1.5
	Winner'11 [30]	23.8±0.8	87.2±1.0
	JS Tiling	**26.5±0.6**	**85.1±0.9**
	Method	MAP(SIFT)	MAP(STIP)
MED	SPM [12]	26.8	17.2
	Winner'12 [29, 21]	27.3	18.7
	JS Tiling	**30.7**	**21.2**
	Method	MAP	-
VOC	SPM [12]	52.5	-
	Winner'07 [15]	54.2	-
	Wang et al. [26]	55.1	-
	Yang et al. [28]	**59.6**	-
	JS Tiling	55.5	-

Table 4: Performance comparison with state-of-the-art methods. The proposed JS Tiling exhibits the best performance on 3 out of 4 datasets. Min DCR is Minimum DCR (the lower the better). MAP(SIFT) and MAP(STIP) indicates the MAP using the SIFT and STIP feature, respectively. "-" denotes the number is unavailable on the dataset.

a similar pattern that the fusion of learned tilings outperforms the SPM at all levels. For example, the proposed JS Tiling improves the MAP of the SPM by 10.7% using four tilings (at $L = 3$) in terms of the MAP. We also compare our best result with the best system in TRECVID 2011 indicated as Winner'11. We conduct experiments using the Winner'11 system [30] and report the performance in Table 4. As we see, JS Tiling improves Winner'11's MAP by a relative 11.3%.

Because of the fixed camera, it is more illustrative to analyze the learned tilings on this dataset. Fig. 4 illustrates the comparison of the best predefined tiling in the SPM with the best learned tiling for two cameras. The heat map plots the distribution of manually annotated bounding boxes of the event in the video. Since localizing events on the whole collection involves a considerable amount of labor, we only annotated two representative events: "PersonRun" and "ObjectPut". The tilings are automatically learned without using the bounding box information. As shown in Fig. 4, the learned tilings seem to be more sensible than the predefined tilings used in the SPM. For example, in Camera 1 (the top part of Fig. 4), the best predefined 2×2 grids intersect the hot region and divide it into four pieces. However, the learned tiling preserves the hot region by merging the tiles around the center. Similarly, the learned tiling in Camera 5 separates the 3 lanes in the airport and tries to preserve the hot region in the middle strip of the video. As the tilings are learned on a coarse-level (tiles), its representation is restricted by the tiles in the mask and thus may not perfectly separate the hot region.

5.3 Multimedia Event Detection (MED)

Our third dataset is the TRECVID MED12 development dataset consisting of the MED12 DEV (2,000 videos) and MED11 DEV-T set (9,746 videos) [21, 29, 9]. The task is to retrieve a ranked list of relevant videos for events such as "Birthday Party" and "Parkour". There are a total of 4,058

Rank	Predefined Masks			Rectangle Masks			All Masks		
	Tiling	Accuracy	MAP	Tiling	Accuracy	MAP	Tiling	Accuracy	MAP
1		79.5±0.7	81.5±0.6		80.4±0.7	83.2±0.6		82.4±0.4	85.5±0.4
2		79.4±0.6	81.8±0.6		80.4±0.4	83.0±0.6		81.4±0.4	84.3±0.5
3		78.6±0.4	80.7±0.4		80.0±0.6	82.4±0.5		80.8±0.5	83.7±0.5
4		77.5±0.2	80.3±0.4		79.9±0.5	82.1±0.7		80.9±0.3	82.5±0.4
5		77.8±0.5	79.6±0.5		79.5±0.7	81.5±0.6		80.4±0.7	83.2±0.6

Table 2: Performance comparison of the best predefined and the best learned tilings on 15-scene categories.

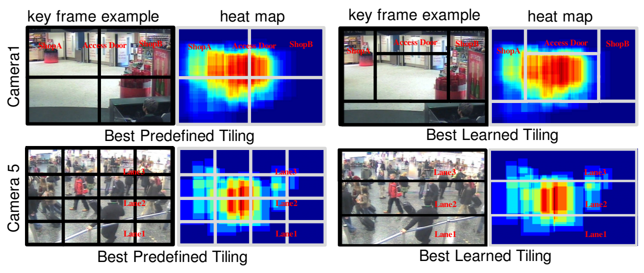


Figure 4: Comparison of the best predefined and learned tilings on Camera 1 (top) and Camera 5 (bottom). The heat map plots the distribution of manually annotated bounding boxes.

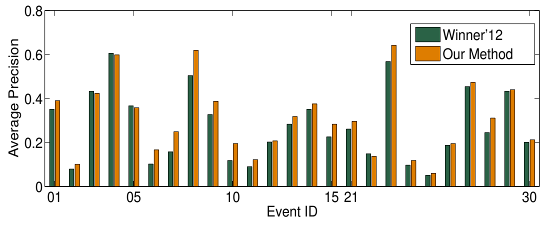
L	Spatial Pyramid		Rectangle Masks		All Masks	
	Min DCR	MAP	Min DCR	MAP	Min DCR	MAP
0	90.1±1.6	22.7±0.9	88.4±1.5	23.5±1.2	87.8±1.3	24.2±1.0
1	89.8±1.2	**23.3±0.7**	87.7±1.0	24.1±0.8	87.6±1.1	23.7±0.6
2	89.2±1.5	23.2±1.0	87.2±1.0	23.8±0.8	86.8±0.9	24.7±0.7
3	**89.0±1.5**	22.8±1.0	**87.1±0.9**	24.3±0.8	86.0±0.9	25.1±0.6
6	-	-	87.3±0.9	**24.5±0.7**	**85.1±0.9**	**26.5±0.6**

Table 5: Comparison of the tiling fusion results on SED. The first column is #tilings used in the fusion. Min DCR denotes Minimum DCR.

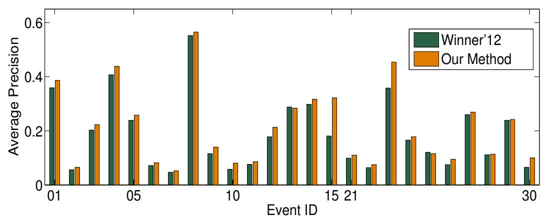
positive videos for the 25 events (E001-E015 and E021-E030) and the remaining 7,688 videos are background videos which do not belong to any of the 25 events. Following the setting in [11, 29], we extract the static image descriptor SIFT [14] and the motion descriptor STIP [25] with the vocabulary size $|V| = 4096$. Table 4 presents the comparison with the baseline method. Winner'12 in Table 4 indicates the best system [21, 29] in the TRECVID 2012 contest according to [17]. We conduct experiments using the system in [29] and report its MAP in the table. Generally, the relative improvements, in terms of MAP, over the best baseline are 12% using SIFT and 13% using STIP. The experiment shows that the learned tilings are applicable to both static image features and motion features. Fig. 5 depicts the comparison between the Winner'12 on each event. As can be seen, JS Tiling outperforms the SPM on 21 out of 25 events using SIFT, and on 23 out of 25 events using STIP.

5.4 Pascal VOC

Following [22], Pascal VOC 2007 is selected as our fourth dataset. The experiments are conducted on the development set called "trainval" (5,011 images) provided by the organizer [7]. We extract SIFT feature with the vocabulary size $|V| = 1024$ and adopt the standard criterion of interpolated MAP used in the challenge [7]. Table 6 presents the comparison between the SPM and the fusion of the learned tilings



(a) SIFT



(b) STIP

Figure 5: Average precision comparison on the MED dataset. The MAP across events is in Table 4.

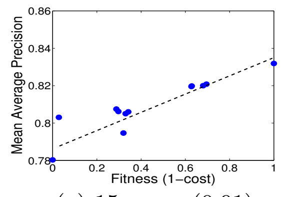
L	Spatial Pyramid	Rectangle Masks	All Masks
	MAP	MAP	MAP
0	49.1±0.2	51.4±0.2	51.4±0.2
1	51.7±0.3	52.8±0.3	52.8±0.2
2	**52.5±0.3**	53.0±0.2	53.0±0.3
3	52.4±0.3	53.6±0.3	53.6±0.4
4	51.6±0.4	53.7±0.4	54.5±0.3
6	-	**54.3±0.3**	**55.3±0.3**

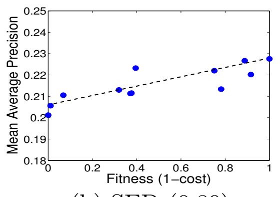
Table 6: Comparison of the tiling fusion results on Pascal VOC.

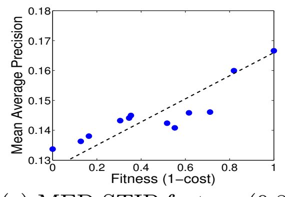
at different levels. As on other datasets, we can observe a similar pattern in which the fusion of learned tilings consistently outperform the SPM across levels. Table 4 presents the comparison with the baseline methods. Note the MAP of all methods in the table are obtained on the same development set "trainval" and Winner'07 denotes the best system in VOC 2007 contest [15]. As we see, JS Tiling improves SP-M by a relative 5.3%. Though it is the second best method in the table, it still shows evident improvement over the SP-M. The best method [28] optimizes the coding dictionary, which is a different optimization direction from this paper. We hypothesize that our performance can be further improved using the better dictionary in [28].

5.5 Efficacy and Efficiency of JS Tiling

To verify the efficacy of Eq. (5), we compare the cost estimated by Eq. (5) with the true classification MAP of a SVM classifier with a χ^2 kernel. Since running classifiers is time consuming, the experiments are conducted on a tractable mask i.e. a 3×3 rectangle mask including 12 equal tilings. Fig. 6 shows the comparison result, where the x-axis represents the tiling's fitness estimated by Eq. (5); the fitness is derived from $1 - cost$ in Eq. (5) and is normalized into $[0, 1]$; the y-axis is the tiling's true MAP of the SVM classifier. The Pearson correlation coefficient of the two values is listed in parentheses under each figure. We can observe a strong correlation between the estimated fitness and the true MAP, with an average coefficient of 0.88 across four datasets. This observation substantiates the hypothesis that Eq. (5) is a reasonably accurate cost function.


(a) 15-scene (0.91)


(b) SED (0.89)


(c) MED STIP feature (0.88)

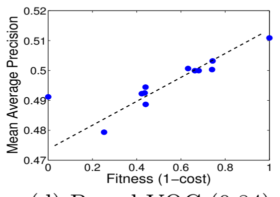

(d) Pascal VOC (0.84)

Figure 6: Classification MAP versus estimated fitness. Fitness is derived from $1 - cost$ in Eq. (5) and normalized to $[0, 1]$. Pearson correlation coefficients are listed in parentheses under each figure.

To study the sensitivity of the parameter λ in Eq. (5) on the classification MAP, we tune λ on the equal tilings generated by a 3×3 rectangle mask. Fig. 7 illustrates the result where the x-axis represents the value of λ and the y-axis is the MAP of the learned tiling. As we see, the MAP is less sensitive to the change of λ, and the performance remains the same in the range of 0.1 to 0.2 across datasets.

To verify the efficiency, we compare the runtime of searching the optimal tiling on all (equal and unequal) tilings generated by a 3×3 rectangle mask using Eq. (5), a linear SVM (including the time for explicit feature mapping), and a SVM with χ^2. The experiments are conducted on a single core Intel Core i7 CPU@2.8GHz with 4G memory. The search space contains 1,434 tilings. The runtime in Table 7 is extrapolated by the runtime on a subspace containing 30 tilings. As we see in Table 7, learning tilings by Eq. (5) is orders of magnitude faster than by kernel and linear SVM. Since the space in the experiment is still very small, in practice, it is computationally infeasible to evaluate tilings with an SVM classifier. The observation confirms the theoretical complexity analysis in Section 4.2 and substantiates the efficiency of the proposed cost function.

6. CONCLUSIONS AND FUTURE WORK

We proposed a novel method called JS Tiling to learn the BoW representation for large-scale datasets directly at the BoW level. JS Tiling offers an approach to design salient spatial BoW representations. The experimental results on four diverse datasets demonstrate that the proposed JS Tiling outperforms spatial pyramid matching and state-of-the-art methods. The runtime comparison further demonstrates that JS Tiling is very efficient and its overhead time of learning tilings is 1,000 times less than that of a conventional method that selects optimal representations by running classifiers.

Empirically, we observed that small tiles that cover a very small area may underestimate the real cost. Because small tiles usually contain a few interest points, and thus are represented by non-smooth BoW distributions. This problem is mainly a result from the "sampling bias" in the cost function. We plan to study smoothing methods to overcome this issue in future.

Dataset	JS Tiling	Linear SVM	Kernel SVM
15-scene	1.1(h)	1,314(h)	10,874(h)
SED	2.1(h)	2,629(h)	32,862(h)
MED	2.3(h)	4,541(h)	41,825(h)
Pascal VOC	1.6(h)	1,912(h)	22,346(h)

Table 7: Runtime comparison of the tiling search (in hours) on the 3×3 rectangle mask.

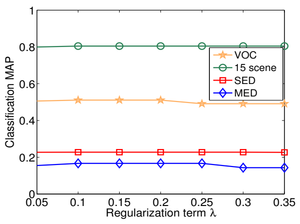


Figure 7: The impact of the regularization term λ in Eq. (5) on the classification MAP.

7. ACKNOWLEDGMENTS

This work was partially supported by Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center contract number D11PC20068. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

8. REFERENCES

[1] JS Tiling webpage. https://code.google.com/p/learning2tile/.
[2] Y. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010.
[3] Y. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In *ICML*, 2010.
[4] C. Chang and C. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.
[5] M. Chen and A. Hauptmann. MoSift: Recognizing human actions in surveillance videos. Technical report, Carnegie Mellon University, 2009.
[6] M. Er. A fast algorithm for generating set partitions. *The Computer Journal*, 31(3):283–284, 1988.
[7] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010.

[8] J. Feng, B. Ni, Q. Tian, and S. Yan. Geometric l_p-norm feature pooling for image classification. In *CVPR*, 2011.

[9] A. Habibian, K. E. van de Sande, and C. G. Snoek. Recommendations for video event recognition using concept vocabularies. In *ICMR*, pages 89–96, 2013.

[10] Y. Jia, C. Huang, and T. Darrell. Beyond spatial pyramids: Receptive field learning for pooled image features. In *CVPR*, 2012.

[11] L. Jiang, A. G. Hauptmann, and G. Xiang. Leveraging high-level and low-level features for multimedia event detection. In *Multimedia*, pages 449–458, 2012.

[12] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[13] X. Li, C. G. Snoek, M. Worring, and A. W. Smeulders. Fusing concept detection and geo context for visual search. In *ICMR*, page 4, 2012.

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

[15] M. Marszalek and C. Schmid. Learning representations for visual object class recognition. In *PASCAL VOC workshop*, 2007.

[16] M. Orlov. Efficient Generation of Set Partitions. Technical report, University of ULM, 2002.

[17] P. Over, J. Fiscus, and G. Sanders. Trecvid 2012–an overview to the goals, tasks, data, evaluation mechanisms, and metrics. In *TRECVID, NIST*, 2012.

[18] G. Sharma and F. Jurie. Learning discriminative spatial representation for image classification. In *BMVC*, 2011.

[19] G. Sharma, F. Jurie, and C. Schmid. Discriminative spatial saliency for image classification. In *CVPR*, 2012.

[20] H. Sharp. Cardinality of finite topologies. *Journal of Combinatorial Theory*, 5(1):82–86, 1968.

[21] W. Tong, Y. Yang, L. Jiang, S. I. Yu, Z. Lan, Z. Ma, W. Sze, E. Younessian, and A. G. Hauptmann. E-LAMP: integration of innovative ideas for multimedia event detection. *Machine Vision and Applications*, pages 1–11, 2013.

[22] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR*, pages 1521–1528. IEEE, 2011.

[23] J. C. van Gemert, C. J. Veenman, A. W. Smeulders, and J. Geusebroek. Visual word ambiguity. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(7):1271–1283, 2010.

[24] V. Viitaniemi and J. Laaksonen. Spatial extensions to bag of visual words. In *ACM CIVR*, 2009.

[25] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, C. Schmid, et al. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009.

[26] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.

[27] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[28] J. Yang, K. Yu, and T. Huang. Efficient highly over-complete sparse coding using a mixture model. In *ECCV*, 2010.

[29] S. I. Yu, Z. Xu, D. Ding, W. Sze, F. Vicente, Z. Lan, Y. Cai, S. Rawat, P. Schulam, N. Markandaiah, et al. Informedia E-LAMP@ trecvid 2012 multimedia event detection and recounting (med and mer). In *TRECVID, NIST*, 2012.

[30] L. Zhang, L. Jiang, L. Bao, S. Takahashi, Y. Li, and A. Hauptmann. Informedia@ trecvid 2011: Surveillance event detection. In *TRECVID, NIST*, 2011.

APPENDIX

LEMMA 1. *Suppose for each tile, its histogram is L1 normalized i.e.* $\forall k$, $\mathbf{D}_+^k = \mathbf{H}_+^k$, $\mathbf{D}_-^k = \mathbf{H}_-^k$. *There are* $|\mathcal{T}_\kappa(S)|$ *tiles in the final representation. The negative JS divergence is approximately the upper bound of the training error of a weighted K Nearest Neighbor classifier, i.e.*

$$-\sum_{k=0}^{|\mathcal{T}_\kappa(S)|-1} JS(\mathbf{D}_k^+ \,\|\, \mathbf{D}_k^-) \geq \frac{1}{8\delta}\sum_i(-y_i\hat{y}_i) + O(\| \mathbf{D}_+^k - \mathbf{D}_-^k \|_3^3) \quad (6)$$

where $\hat{y}_i = \sum_j y_j(\frac{\mathbf{x}_i}{\delta_i})^T\frac{\mathbf{x}_j}{\delta_j}$ *calculates the estimated label for the ith sample.* δ *is a positive constant.* $O(\| \mathbf{D}_+^k - \mathbf{D}_-^k \|_3^3)$ *is the higher order polynomials in Taylor's series. An alternative form of the training error is given by Eq.* (17).

PROOF. Expand $\log a$ at the point b by Taylor's theorem:

$$\log a - \log b = (a-b)\frac{1}{b} - \frac{1}{2b^2}(a-b)^2 + \frac{1}{6\epsilon^3}(a-b)^3 \quad (7)$$

where ϵ is a constant between a and b. According to the definition in Eq. (4).

$$JS(\mathbf{x} \,\|\, \mathbf{y}) = \frac{1}{2}\sum_{t=1}^{|V|} x(t)\log\frac{2x(t)}{x(t)+y(t)} + y(t)\log\frac{2y(t)}{x(t)+y(t)} \quad (8)$$

Substitute Eq. (7) into Eq. (8), we have:

$$JS(\mathbf{x} \,\|\, \mathbf{y}) = \sum_{t=1}^{|V|}\frac{[x(t)-y(t)]^2}{4[x(t)+y(t)]} + \frac{1}{12}[x(t)-y(t)]^3(\frac{x(t)}{\epsilon_1^3} - \frac{y(t)}{\epsilon_2^3}) \quad (9)$$

According Eq. (2), $x(t)$ and $y(t)$ corresponds a dimension in the BoW distribution after Laplace (adding α) smoothing. We have $\forall t$, $0 < \delta \leq x(t), y(t) < 1$, where δ is a positive constant. According to Taylor's theorem, $\delta < \epsilon_1$, $\epsilon_2 < 1$. Since $x(t) + y(t) \geq 2\delta$, we have

$$JS(\mathbf{x} \,\|\, \mathbf{y}) \leq \frac{\| \mathbf{x} - \mathbf{y} \|_2^2}{8\delta} + \frac{\| \mathbf{x} - \mathbf{y} \|_3^3}{12\delta^3}. \quad (10)$$

The second term is the higher order polynomials in Taylor's series which can be represented as $O(\| \mathbf{x} - \mathbf{y} \|_3^3)$.

For the k^{th} tile in the mask $\mathcal{T}_\kappa(S)$, we have:

$$\begin{aligned}\| \mathbf{H}_+^k - \mathbf{H}_-^k \|_2^2 = \sum_{t=1}^{|V|}(\sum_{y_i=+1}\frac{x_i^k(t)}{n^+})^2 \\ + \sum_{t=1}^{|V|}(\sum_{y_j=-1}\frac{x_j^k(t)}{n^-})^2 - 2\sum_{t=1}^{|V|}(\sum_{y_i=+1}\frac{x_i^k(t)}{n^+}\sum_{y_j=-1}\frac{x_j^k(t)}{n^-})\end{aligned} \quad (11)$$

Note that, using the dot product trick, we have:

$$\begin{aligned}\sum_{t=1}^{|V|}(\sum_{y_i=+1}\frac{x_i^k(t)}{n^+})^2 = \sum_{y_i,y_j=+1}\frac{{\mathbf{x}_i^k}^T\mathbf{x}_j^k}{{n^+}^2} \\ \sum_{t=1}^{|V|}(\sum_{y_i=+1}\frac{x_i^k(t)}{n^+}\sum_{y_j=-1}\frac{x_j^k(t)}{n^-}) = \sum_{y_i=+1}\sum_{y_j=-1}\frac{{\mathbf{x}_i^k}^T\mathbf{x}_j^k}{n^+n^-}\end{aligned} \quad (12)$$

and

$$\sum_{k=0}^{|\mathcal{T}_\kappa(S)|-1}(\mathbf{x}_i^k)^T\mathbf{x}_j^k = (\mathbf{x}_i)^T\mathbf{x}_j \quad (13)$$

Let δ_i denote an indicator function which equals n^+ when $y_i = +1$ and n^- when $y_i = -1$. Based on Eq. (12) and Eq. (13), we can rewrite Eq. (11) as

$$\sum_{k=0}^{|\mathcal{T}_\kappa(S)|-1}\| \mathbf{H}_+^k - \mathbf{H}_-^k \|_2^2 = \sum_{i,j}(y_iy_j)(\frac{\mathbf{x}_i}{\delta_i})^T\frac{\mathbf{x}_j}{\delta_j} \quad (14)$$

Substitute Eq. (14) back into Eq. (10), we have:

$$-\sum_{k=0}^{|\mathcal{T}_\kappa(S)|-1} JS(\mathbf{H}_+^k \,\|\, \mathbf{H}_-^k) \geq -\frac{1}{8\delta}\sum_{i,j}(y_iy_j)(\frac{\mathbf{x}_i}{\delta_i})^T\frac{\mathbf{x}_j}{\delta_j} + O(\| \mathbf{H}_+^\mathbf{k} - \mathbf{H}_-^\mathbf{k} \|_3^3) \quad (15)$$

Let $\hat{y}_i = \sum_j y_j(\frac{\mathbf{x}_i}{\delta_i})^T\frac{\mathbf{x}_j}{\delta_j}$ and using the condition $\forall k$, $\mathbf{D}^k = \mathbf{H}^k$

$$-\sum_{k=0}^{|\mathcal{T}_\kappa(S)|-1} JS(\mathbf{D}_+^k \,\|\, \mathbf{D}_-^k) \geq \frac{1}{8\delta}\sum_i(-y_i\hat{y}_i) + O(\| \mathbf{D}_+^\mathbf{k} - \mathbf{D}_-^\mathbf{k} \|_3^3) \quad (16)$$

To make it clearer, plus a constant $(n^+ + n^-)/8\delta$ on the both sides we have:

$$\begin{aligned}\frac{1}{8\delta}(n^+ + n^-) - \sum_{k=0}^{|\mathcal{T}_\kappa(S)|-1} JS(\mathbf{D}_k^+ \,\|\, \mathbf{D}_k^-) \\ \geq \frac{1}{8\delta}\sum_i(1 - y_i\hat{y}_i) + O(\| \mathbf{D}_+^\mathbf{k} - \mathbf{D}_-^\mathbf{k} \|_3^3)\end{aligned} \quad (17)$$

where $\hat{y}_i$ calculates the label for ith training sample according to all of samples in the dataset. The right-hand side term of Eq. (16) calculates the training error for the ith sample by a linear loss function. Therefore the negative JS divergence is the upper bound of the training error of a weighted K-NN classifier when $K = n^+ + n^-$. $\square$