Densified Winner Take All (WTA) Hashing for Sparse Datasets

Beidi Chen
Computer Science Dept.
Rice University
beidi.chen@rice.edu

Anshumali Shrivastava
Computer Science Dept.
Rice University
anshumali@rice.edu

Abstract

WTA (Winner Take All) hashing has been successfully applied in many large-scale vision applications. This hashing scheme was tailored to take advantage of the comparative reasoning (or order based information), which showed significant accuracy improvements. In this paper, we identify a subtle issue with WTA, which grows with the sparsity of the datasets. This issue limits the discriminative power of WTA. We then propose a solution to this problem based on the idea of Densification which makes use of 2-universal hash functions in a novel way. Our experiments show that Densified WTA Hashing outperforms Vanilla WTA Hashing both in image retrieval and classification tasks consistently and significantly.

1 INTRODUCTION

In many important applications like information retrieval and natural language processing, text documents, and images data are in high-dimensional representations. Such high-dimensionality is usually accompanied by extreme data sparsity due to either a large vocabulary or the use of large image window size. The major reason we find very sparse datasets almost everywhere results from the wide adoption of Bag of Words (BoW) representation for documents and images. In BoW representation, the presence or absence of specific features carries the most information [Chapelle et al., 1999, Jiang et al., 2007], especially with higher order shingles. The popularity of sparse machine learning [Caiafa et al., 2017, Liu and Tsang, 2017, Liu et al., 2017, Liu and Tsang, 2016] and sparse codes [Lee et al., 2006] for image data is another reason for the abundance of sparse datasets in modern applications. In order to get a sense of this extreme sparsity, the datasets demonstrated in Google's Machine Learning system SIBYL [Canini et al., 2012] have dimensions in billions and non-zeros in only a few thousands (even hundreds).

With the advent of the Internet and the explosion in volumes of data, almost all machine learning and data mining applications are constrained by their computational requirements. Learning with none-liner kernels, by materializing kernel matrices, which are quadratic in computation and memory, is infeasible [Rahimi and Recht, 2007, Li et al., 2011, Shrivastava, 2015]. Randomized algorithms, especially those based on Locality Sensitive Hashing (LSH) [Indyk and Motwani, 1998], have shown huge promise for reducing computational and memory requirement in these scenarios. These randomized algorithms lead to drastic gains in computation and memory for a small, insignificant, amount of approximations.

LSH-based algorithms are quite popular efficient sublinear algorithms for near neighbor search [Indyk and Motwani, 1998]. This is because even a simple linear scan for near neighbor search, over massive datasets, becomes prohibitively expensive [Weber et al., 1998]. There are no options but to use hashing approaches for such scenarios. LSH algorithms can also be used as cheap random kernel features [Li et al., 2011] for training large-scale non-linear SVMs without materializing the expensive kernel matrix, leading to linear time algorithms. Besides, recently a line of work appears to use LSH as samplers in optimization [Chen et al., 2018] and deduplication [Chen et al., 2017] problems. They are embarrassingly parallel, simple and cheap. Owing to these unique advantages, they are heavily used by commercial search industries for truly large-scale data processing systems.

In the last decade, similarities based on relative (or comparative) attributes have gained huge popularity, especially in the vision literature [Parikh and Grauman, 2011]. For such similarities, a well-known hashing

scheme is *Winner Take All (or WTA) hashing* [Yagnik et al., 2011]. **It is one of the fastest known hashing scheme, which is much faster than signed random projection (SRP)**. SRP requires one pass over the data vector for computing one hash value. This is expensive because in practice we need hundreds of hash values, which results in hundreds of passes over the data. Similarly, even random projections are significantly slow for many large-scale tasks. On the contrary, WTA can generate multiple hashes in one pass. It is widely known that hashing time is the major bottleneck, both in theory and practice, for the task of image retrieval. This is why Google [Dean et al., 2013] needed WTA for detecting 100,000 objects on a single machine in near-real time with very respectable accuracy.

Large-scale image retrieval, with low-latency constraints, is a reality. We cannot afford to have costly hash functions since even one pass over the data vector for hash computation is prohibitively expensive both for energy and latency. WTA hashing has been quite successfully applied to produce superior results on massive-scale object recognition and information retrieval. This randomized hashing scheme seems quite suitable for taking advantage of multiple partial order statistics rather than total orderings of the input vector's feature dimensions to produce sparse embedding codes.

Deep Neural Networks are widely-used in vision and speech tasks. While the network architecture sizes grow exponentially larger to adapt data complexity, LSH algorithms are recently adopted to reduce the computation [Spring and Shrivastava, 2017, Vijayanarasimhan et al., 2014]. Moreover hashing cost and quality are the critical bottleneck in making such approaches practical.
Our Contributions: In this work, we study the applicability of WTA hashing for very sparse datasets. We found that WTA hashes are not very informative for sparse datasets. We further provide a remedy based on the recent idea of *Densification* [Shrivastava and Li, 2014a]. In particular, our contributions can be summarized as follows:

1. We illustrate that the popular WTA hashing scheme starts losing information for very sparse datasets, i.e., most of the hash values for very sparse datasets do not have enough discriminative information.

2. We propose Densified WTA Hashing which combines traditional WTA hashing with the idea of Densification [Shrivastava, 2017]. We show that the idea of densification provably fixes the issue of WTA for sparse datasets. Our proposal makes novel use of 2-universal hashing, introduced in Section 4.1, and requires minimal modifications to the original WTA hashing. Furthermore, for dense datasets, our proposal is equivalent to the original WTA hashes and thus a smooth generalization of WTA for sparse datasets.

3. We show for the first time that the idea of Densification actually leads to significant improvement in the quality of WTA hashing, informative hashes. Previously the idea of Densification was only known to speed up hash functions without losing quality. Furthermore, this is the first use of densification for non-binary data.

4. We demonstrate the benefits of our proposal by showing significant gains in accuracy compared to WTA on real-world sparse datasets for both retrieval and classification tasks.

2 REVIEW WTA HASHING

[Parikh and Grauman, 2011] pointed out the importance of relative attributes in the vision community. It suggested that for a given vector x, the information that the attribute x_i is dominant over some other attribute x_j has stronger discriminative powers compared to other features. It was further shown in [Yagnik et al., 2011] that comparative reasoning (or order information) among attributes is a very informative feature and similarities based on such comparisons lead to superior performances compared to widely adopted measures like L_2 distances. However, kernel based (or similarity based) learning is computationally slow. To mitigate this problem, WTA (Winner Takes ALL) Hashing was proposed. The simplicity, scalability, and power of WTA hashing were quite appealing and it has been successfully used by commercial big-data companies to scale up the task of object detection significantly [Dean et al., 2013].

WTA hashing generates a set of random samples of K attributes, using a random permutation Θ, and stores the index of the attribute with the maximum weight. It can be implemented in three lines with Matlab:

```
function [maxval,c] = wta(X,K)
theta = randperm(size(X,2))
[maxval,c] = max(X(:,theta(1:K)),[],2)
```

2.1 KEY WTA NOTATIONS

We denote $\Theta(x)$ to be the K random samples from x sampled using permutation Θ. For convenience, we drop the dependence on K as it will remain a fixed constant. $\mathcal{H}_{wta}(\Theta(x))$ indicates the corresponding WTA hash. We will also drop Θ and use $\mathcal{H}_{wta}(x)$ when it is clear.

As illustrated in the example shown in Table 1, the original input vectors x_1, x_2, x_3, x_4 are applied with random

Table 1: WTA Hashing Example with four input vectors x_1, x_2, x_3, x_4, $K = 3$ and one permutation $\Theta = 4, 1, 2$

	$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{x_4}$
$\mathbf{x}$	**10, 12, 9, 23**	**8, 9, 1, 12**	**9, 2, 6, 1**	**3, 5, 1, 7**
$\mathbf{\Theta(x)}$	**23, 10, 12**	**12, 8, 9**	**1, 9, 2**	**7, 3, 5**
$\mathcal{H}_{\mathbf{wta}}(\mathbf{x})$	**1**	**1**	**2**	**1**

permutation $\Theta = (4, 1, 2, 3)$ and first $K = 3$ attributes of the permuted vectors are selected (random sample of size 3), e.g. Vector (a) $= [10, 12, 9, 23]$ will sample $[23, 10, 12]$. Then the index of the maximum attribute in every transformed vector is stored separately, e.g. 1 for (a), to contribute to the final WTA hash codes. If there are n such hashes codes for one input vector, **we define Bin i as the space to store the hash code generated from the i^{th} set of K samples.**

It was shown that WTA hashing scheme has locality sensitive hashing property. It implies that collision probability under this scheme, i.e. for given vectors x and y, $Pr(\mathcal{H}_{wta}(x) = \mathcal{H}_{wta}(y)) = \mathbb{E}[\mathbf{I}_{\mathcal{H}_{wta}(x)=\mathcal{H}_{wta}(y)}]$ is some desirable order based similarity measure. It was later shown that for $K = 2$ this similarity is the well known Kendall Tau [Ziegler et al., 2012].

3 SPARSE DATASETS AND ISSUES WITH WTA HASHING

WTA hashing and the idea of comparative reasoning is quite appealing and intuitive. In this section, we delve deeper and show a critical issue with WTA hashing. We show that for very sparse datasets, which are common in practice [Li et al., 2011], WTA-based hashes are not very informative and deviate from the "relative attribute" intuition. We use the equivalence between hashing and the kernel view to illustrate this issue. With every hashing scheme $\mathcal{H}$ is an associated positive definite kernel given by the collision probability $Pr(\mathcal{H}(x) = \mathcal{H}(y)) = \mathbb{E}[\mathbf{I}_{\mathcal{H}(x)=\mathcal{H}(y)}]$. For large-scale learning, as shown in [Yagnik et al., 2011], we can convert these hashes into random kernel features [Rahimi and Recht, 2007] by converting hash values to indicator vectors.

3.1 SPARSITY MAKES WTA UNINFORMATIVE

Define the sparsity of a dataset X with n samples, with each sample of dimension d, as

$$S_x = \frac{\sum_{i=1}^{n} \sum_{j=1}^{d} [1\{X_{ij} = 0\}]}{n \times d} \tag{1}$$

Note that $[1\{X_{ij} = 0\}]$ is an indicator for the event $X_{ij} = 0$. S_x is also the probability that $Pr(X_{ij} = 0)$. We will show that the kernel associated with WTA hashing becomes uninformative as the sparsity increases.

Consider the example that is shown in Table 2. Given very sparse input vectors x_1, x_2, we generate six WTA hashes with $K = 3$. In order to do this, we sample $K = 3$ attributes six different times so that each different bin is generated using a different permutation. Due to sparsity, many of these bins contain all zeros. We can see that in all the bins except Bin 5, $\mathcal{H}_{wta}(x_1)$ and $\mathcal{H}_{wta}(x_2)$ collide and therefore the estimated collision probability, from the hashes, is roughly $\frac{5}{6}$ indicating high similarity (1 is maximum). This seems misleading.

Due to sparsity, it is very likely that for a given x, all the sampled attributes $\Theta(x)$ are zeros for some samples. We represent this situation by $\Theta(x) = E$ (Empty). Consider Bin 1, 4 and 6, they collide only because they are all zeros. Note, WTA treats all empty bins as collisions and two empty bins will always lead to a hash collision. Sparse datasets are common with Bag-of-Words (or token-based) representation. Empty Bins (1, 4 and 6) indicate the absence of the randomly chosen K tokens which is not a strong indicator of similarity. In BoW analogy, if two documents concurrently lack the words "Hashing", "Winner" and "Take", it does not indicate strong similarity given the large vocabulary and the sparse nature of the dataset. In sparse BoW representation, the absence of features is not informative but only the presence of features is important. Thus, whenever the bins in both input vectors, under considerations for WTA, are empty, we observe undesirable collisions.

However, it is also problematic if we treat empty ones as mismatches. For two identical sparse vector, ideally they should always collide as they are identical. But if we treat zeros as mismatches, then even identical vectors would have low collision probability. If hashes do not collide, it is an indicator that the input vectors are not similar. Preventing empty bins from colliding will treat sparsity as dissimilarity, which is again undesirable. Thus, there is no straightforward fix to this problem.

If we further observe Bin 3, the collision is even worse because it is meaningless that an empty Bin of x_2 collides with a non-empty bin of x_1, simply because the max value in x_1 happens to be at index 1. This is actually a spurious collision and can be easily eliminated if we assign special values to all empty bins. Therefore, from the analysis, we ignore this easily fixable but spurious collision.

Table 2: WTA with input vectors x_1, x_2 and six bins generated with six permutations. E denoted an empty sampling. WTA treats E and E as a match of hash values, which artificially inflates the similarity perceived by the hashes.

$\mathbf{x_1}$	**0, 0, 5, 0, 0, 7, 6, 0, 0**
$\mathbf{x_2}$	**0, 0, 1, 0, 0, 0, 0, 0, 0**

	Bin 1	**Bin 2**	**Bin 3**	**Bin 4**	**Bin 5**	**Bin 6**
$\mathbf{\Theta}$	**2, 1, 8**	**5, 3, 9**	**6, 2, 4**	**8, 9, 1**	**1, 7, 3**	**2, 4, 5**
$\mathbf{\Theta(x_1)}$	**0, 0, 0 (E)**	**0, 5, 0**	**7, 0, 0**	**0, 0, 0 (E)**	**0, 6, 5**	**0, 0, 0 (E)**
$\mathbf{\Theta(x_2)}$	**0, 0, 0 (E)**	**0, 1, 0**	**0, 0, 0 (E)**	**0, 0, 0 (E)**	**0, 0, 1**	**0, 0, 0 (E)**
$\mathcal{H}_{\mathbf{wta}}(\mathbf{x_1})$	**1 (E)**	**2**	**1**	**1 (E)**	**2**	**1 (E)**
$\mathcal{H}_{\mathbf{wta}}(\mathbf{x_2})$	**1 (E)**	**2**	**1 (E)**	**1 (E)**	**3**	**1 (E)**

In Bin 2, neither $\Theta(x_1)$ nor $\Theta(x_2)$ are E, so those are informative collisions. This is in line with the original motivation of WTA. Owing to the presence of empty bins, sparsity dominates the hash representations of x_1, x_2 and leads to high undesirable similarity. We can not simply ignore empty values because different vectors will have different occurrences of empty bins. Please refer [Shrivastava and Li, 2014a] to see in details why there is no way to ignore empty values in indexing.

Formally, given vectors x_1, x_2 and a permutation Θ, define the indicator vector for empty sampling of both x_1 and x_2:

$$I_{empty} = \begin{cases} 1 & \Theta(x_1) = \Theta(x_2) = E \\ 0 & otherwise \end{cases} \quad (2)$$

Note if any of the $\Theta(x_1)$ is not empty then $I_{empty} = 0$. Based on this indicator variable, we can define empty and non-empty collisions as:

$$k_{bad}(x_1, x_2) = Pr(\mathcal{H}_{wta}(x_1) = \mathcal{H}_{wta}(x_2) | I_{empty} = 1)$$
$$k_{good}(x_1, x_2) = Pr(\mathcal{H}_{wta}(x_1) = \mathcal{H}_{wta}(x_2) | I_{empty} = 0).$$

As argued, $k_{bad}(x_1, x_2)$ is not an informative kernel for very sparse datasets. Using these quantities we can formally write the WTA kernel as

$$\begin{aligned} k_{wta}(x_1, x_2) &= Pr(\mathcal{H}_{wta}(x_1) = \mathcal{H}_{wta}(x_2)) \quad (3) \\ &= a k_{bad}(x_1, x_2) + (1 - a) k_{good}(x_1, x_2), \end{aligned}$$

where a is the probability of $I_{empty} = 1$. Clearly, for very sparse datasets a will be high and hence $k_{bad}(x_1, x_2)$ dominates the WTA kernel making it less discriminative.

4 OUR PROPOSAL: DENSIFIED WTA HASHING

4.1 2-UNIVERSAL HASHING

Definition 1. *A randomized function $h_u : [l] \rightarrow [k]$ is 2-universal if, $\forall i, j \in [l]$ with $i \neq j$, we have the following property for any $z_1, z_2 \in [k]$*

$$Pr(h_u(i) = z_1 \text{ and } h_u(j) = z_2) = \frac{1}{k^2}. \quad (4)$$

A simple universal hash function example would be, for random number a and b and a prime number $p \leq k$, compute: $h_u(x) = (ax + b \bmod p) \bmod k$.

4.2 PROPOSAL

In [Shrivastava and Li, 2014b] the authors proposed the idea of Densification of hashes for obtaining a one-pass hashing scheme which has the same collision probability as the traditional minwise hashing. The idea was to reassign empty bins, having all zero values, by borrowing values from nearest non-empty bins added with some constant offset. Furthermore, [Shrivastava, 2017] showed a better densification schema with optimal variance. Motivated by this idea, we propose a similar reassignment of empty bins generated from WTA. We will show that the modified WTA, which we call "Densified WTA" (DWTA) hashing, produces the right kernel. This is little surprising because *Densification* was used in the literature to speed up the hashing scheme with the same old property. Here we rather show a first example where densification improves the hashing scheme by making it more informative. This is also the first use of densification over non-binary data.

Vanilla WTA assigns all empty bins a constant value of 1. Using densification, we assign new random values to all the empty bins. For a given data vector x, we first generate a set of WTA hashes and place them one after the other (See Table 3).

The overall procedure of Densification for reassigning the empty bins is shown in Algorithm 1. We do not touch non-empty bins, as we know that WTA hashes are informative enough. Thus, if a bin is non-empty, its WTA hash value is the DWTA hash value. The key idea in this algorithm is that when a bin i is empty, instead of assigning it with a constant 1 like what WTA does, it chose

Table 3: Example densification of WTA hashes shown in Table 2. All the hash values of empty bins are reassigned (shown in red) by the values of the mapped (using $h_u(.,.)$ and lookup table in Table 4) non-empty bins with offset shown by the arrow. This unusual procedure actually is the right fix for WTA as shown by Theorem 1

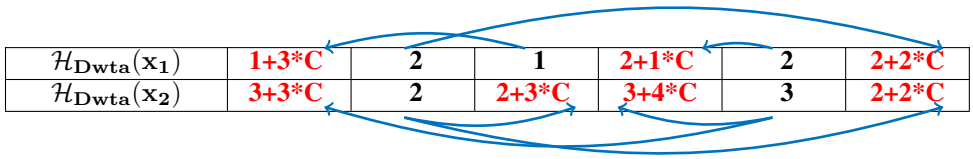

Table 4: Results of empty bins re-assignment mapping in Table 2 running Algorithm 1. i and attempt are the two arguments for some 2-universal hash function and map represents the non-empty bin i is mapped to.

	i	attempt	map
x_1	1	3	3
	4	1	5
	6	2	2
x_2	1	3	5
	3	3	2
	4	4	5
	6	2	2

some non-empty bin randomly using a 2-universal hash function, h_u and use the value of the chosen non-empty bin with some appropriate offset that ensures no spurious collisions. The 2-universal hash function takes in two arguments: 1) the index of the current empty bin and 2) the number of attempts to reach the first non-empty bin. The first argument is to ensure that DWTA will produce good kernels defined in Section 3. Specifically, for instance, in Table 2, Bin 1s are both empty for x_1 and x_2. The ideal collision probability of such empty bins should be the same as that of two non-empty bins, derived in Equation 3. The second argument, $attempts$, is to prevent infinite cycles during the process of reaching the non-empty bin. For instance, when we compute the non-empty bin mapping for Bin i, if h_u only takes in i as an argument and $i = h_u(i)$, then the algorithm would run into an infinite loop. However, with such monotonically increasing $attempts$, even under the same i, the sequence of hash values generated from h_u will not run into infinite cycles. Another scenario that can test the randomness of our algorithm is when $j = h_u(i, attempt)$ and bin i and j are both empty. Under such circumstance, bin i and j are not guaranteed to be re-assigned with the value of the same bin because the re-assignments are independent due to 2-universality of $h_u(.,.)$.

For each empty bin i, we locate a random (but consistently chosen) non-empty bin j according to a 2-universal hash function, call it h_u. Formally,

$$\mathcal{H}_{Dwta}[i] = \mathcal{H}_{wta}[j] + attempt * C. \quad (5)$$

Algorithm 1 Densified WTA Hashing

input n hashes $\mathcal{H}_{wta}[]$ generated from WTA Hashing
input $h_u(.,.)$, constant C
Initialize $\mathcal{H}_{Dwta}[] = 0$
for $i = 1$ to n do **do**
  if $\mathcal{H}_{wta}[i] \neq E$ **then**
    $\mathcal{H}_{Dwta}[i] = \mathcal{H}_{wta}[i]$
  else
    $attempt = 1$
    $next = h_u(i, attempt)$
    while $\mathcal{H}_{wta}[next] = E$ **do**
      $attempt++$
      $next = h_u(i, attempt)$
    end while
    $\mathcal{H}_{Dwta}[i] = \mathcal{H}_{wta}[next] + attempt * C$
  end if
end for
return $\mathcal{H}_{Dwta}[]$

Then the newly assigned value to the empty bin i is exactly the value of j with some appropriate offset. The offset is mainly the number of attempts such process make before termination, multiplying by some constant $C > K$. Table 3 gives a toy example of how Algorithm 1 works on table 2. For x_1, from Table 4, the mapped non-empty bin for Bin 1 with map function h_u is 3 and Bin 3's hash value is 1. The total attempts made for reaching Bin 2 is 3. Therefore, according to equation 5, the new hash value of Bin 1 would be $1 + 3 * C$. Similarly, Bin 4 is assigned with $2 + 1 * C$ and Bin 6 is assigned with $2+2*C$. Reassignments in the same manner happen to x_2 but since it is more sparse than x_1, more bins are filled with new hash values. Recall in Issues with WTA Hashing Section, we discuss that the collisions between $\mathcal{H}_{wta}(x_1)$ and $\mathcal{H}_{wta}(x_2)$ happened in Bin 1, 4 and 6. After densification, there is no collision in Bin 1 and 4. Therefore after densification the hash collision similarity comes down to $\frac{2}{6} = 0.33$.

Formally, let us assume that we want to generate n hash values. $\Theta_i(x)$ denote bin i. Let $h_u(i, attempt)$ be the first number in the process decribed in Algorithm 1 such that $\Theta_{h_u(i,attempt)}(x) \neq E$. We can define the Densified

Table 5: Each entry displays the Sparsity of VOC2010, LabelMe-12-50k and MSRc datasets in 1000 BoW, 5000 BoW and 10000 BoW representation. Sparsity shows the Raw Data sparsity of original BoW vectors and Empty Codes shows the ratio of empty hash codes in resulting WTA Hashing encoding (empty codes means empty sampling). By increasing dictionary size, Sparsity naturally goes up in all three datasets.

	1000 BoW (%)		**5000 BoW** (%)		**10000 BoW** (%)	
	Sparsity	**Empty Codes**	**Sparsity**	**Empty Codes**	**Sparsity**	**Empty Codes**
VOC2010	68.63	23.84	88.18	61.39	92.87	74.81
LabelMe-12-50k	58.07	13.63	82.93	48.18	89.49	64.43
MSRc	69.46	24.66	86.83	56.60	91.54	70.07

WTA, $\mathcal{H}_{Dwta}$, as follows

$$\mathcal{H}_{Dwta}(\Theta_i(x)) = \begin{cases} \mathcal{H}_{wta}(\Theta_i(x)) & \text{if } \Theta_i(x) \neq E \\ \mathcal{H}_{wta}(\Theta_{h_u(i,attempt)}(x)) + attempt * C & otherwise. \end{cases} \quad (6)$$

Based on this definition, we now show our main result that $\mathcal{H}_{wta}$ precisely fixes the issue of empty bins and get rid of the bad kernels. Since the result holds for any bin, we will drop the subscript i. Formally,

Theorem 1. *For any given x and y, the collision probability of "Densified WTA" $\mathcal{H}_{Dwta}$ satisfies:*

$$Pr(\mathcal{H}_{Dwta}(x_1) = \mathcal{H}_{Dwta}(x_2)) = k_{good}(x_1, x_2) = k_{Dwta}(x_1, x_2), \quad (7)$$

Proof: See supplementary material. □

From Theorem 1, it is clear that the new kernel is precisely the good kernel $k_{good}(x_1, x_2)$ with no contribution of $k_{bad}(x_1, x_2)$ in $k_{Dwta}(x_1, x_2)$, irrespective of the sparsity.

4.3 COST OF DENSIFICATION

We can see that we incur an additional cost of densification over the generated WTA hashes. The cost comes from, as shown in Algorithm 1, if the bin is empty, it requires an additional while loop. Let n be the total number of bins in $\mathcal{H}_{Dwta}(\Theta(x))$ and n_{NE} be the number of non-empty bins. The probability of terminating the while loop in one iteration is $p = \frac{n_{NE}}{n}$. Therefore the expected iterations each while loop need to run before termination will be $\frac{1}{p}$. The computation is negligible because it only involves $\frac{1}{p}$ hash lookups for every empty bin. We will show in Section 5.4 that this negligible cost leads to huge performance gains in practice. This we believe is one of the many examples where a careful analysis and some mathematics goes a long way in designing simple and significantly better algorithms.

4.4 DEALING WITH LARGE HASH VALUES

It can be seen from Equation 6 that the value of $\mathcal{H}_{Dwta}(\Theta_i(x))$ can become large due to the term $attempt*C$. It turns out that this is not a problem. There is a significant amount of literature to reduce the final range of hashing scheme [Li and König, 2011]. The idea is to randomly shrink the range at an insignificant cost of small constant random collisions. We found that if we want to constrain the final hash value to a range R simply taking mod R of the final hash value suffices in practice. This is what we use during evaluations.

5 EXPERIMENTS

In this section, we compare the performance of Densified WTA hashing with Vanilla WTA on two tasks: 1) Image retrieval and 2) classification. *The experiments do not compare with other hashing algorithms because the goal of this paper is solving the problem of WTA while maintaining its superiority over other methods mentioned in the introduction section*. They are important tasks of evaluating the performance of Hashing algorithms, because hashing has received increasing interests in efficient large-scale image retrieval with the rapid growth of web images and the classification accuracy can quantify the discriminative power in hashes.

5.1 DATASETS AND BASELINES

We use three popular publicly available image datasets, including VOC2010 [Everingham and Winn, 2010], LabelMe-12-50k [Russell et al., 2008] and MSRc [msr, 2004]:

- The VOC2010 database contains a total of 10103 annotated images of twenty classes, including people, animals, vehicles and indoors. The data has been split into 50% for training and 50% for testing. One image could belong to different classes.
- The LabelMe-12-50k dataset consists of 50,000

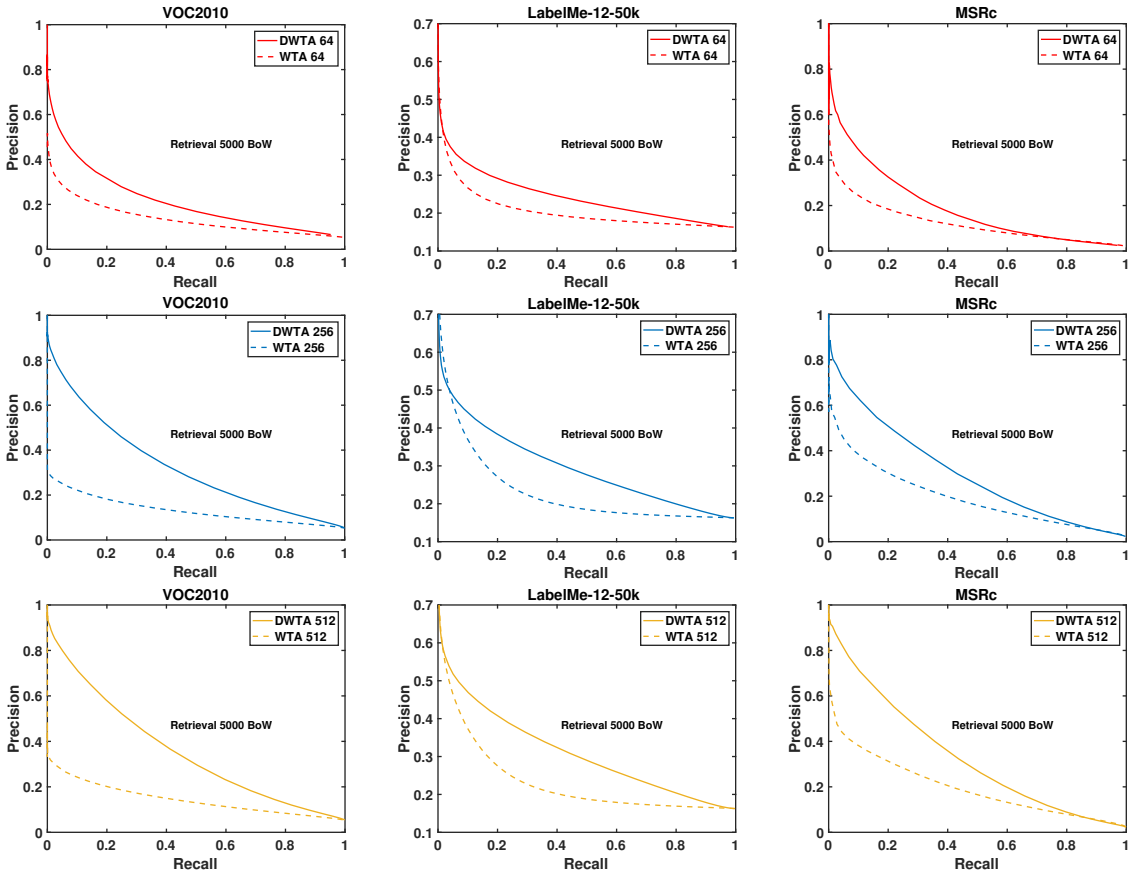

Figure 1: Precision and Recall curves comparing the retrieval performance of Densified WTA vs. WTA on VOC2010, LabelMe-12-50k and MSRc datasets for 1000, 5000 and 10000 BoW feature representations. The semi-dotted lines are the vanilla WTA hashes and bold lines are our proposed Densified WTA Hashes. Different colors represent different number of BoW. We only show 5000 BoW with 64, 256 and 512 hashes (number of hashes used for ranking). Densified WTA significantly outperforms the corresponding WTA consistently.

JPEG images of twelve classes, 80% for training and 20% for testing. They are 256×256-pixels pictures extracted from LabelMe.

- The MSRc is a database of thousands of labeled, high-resolution (680x480 pixels) images of eighteen classes.

The authors of WTA paper used LabelMe for retrieval tasks and VOC2010 datasets for classification tasks. We demonstrate both retrieval and classification on both of the datasets as well as a new MSRc dataset. As described in Section of Large Hash Values, to reduce the space of Densified WTA Hashing, we apply mod operation on hash values of all bins as a fix. Table 5 summarizes the sparsity of Raw Data, input Bag of Words, and the ratio of Empty Hash codes, the resulting codes after applying WTA Hashing to input Bag of Words vectors. We can see that when the number of BoW increases, sparsity, highest in 10000 BoW, also goes up in all three datasets. Note here, we are doing the same tasks as WTA paper, but we do not apply exactly same settings and the sparsities of BoW would thereby be different (they did not reveal sparsity of their datasets as well). Therefore, we do not expect the same results on VOC2010 dataset due to the sparsity difference.

5.2 IMAGE RETRIEVAL

We now compare the performance of our Densified WTA codes with Vanilla WTA by replicating the retrieval experiments and studying the standard precision-recall curves. This is our main task of performance comparison because like we mentioned in the Introduction section, WTA is quite appealing for information retrieval. We restress that WTA (and our DTWA) are the fastest known hashing scheme, significantly faster than plain random projections. Furthermore hashing cost is a critical bottleneck in large-scale retrieval system.

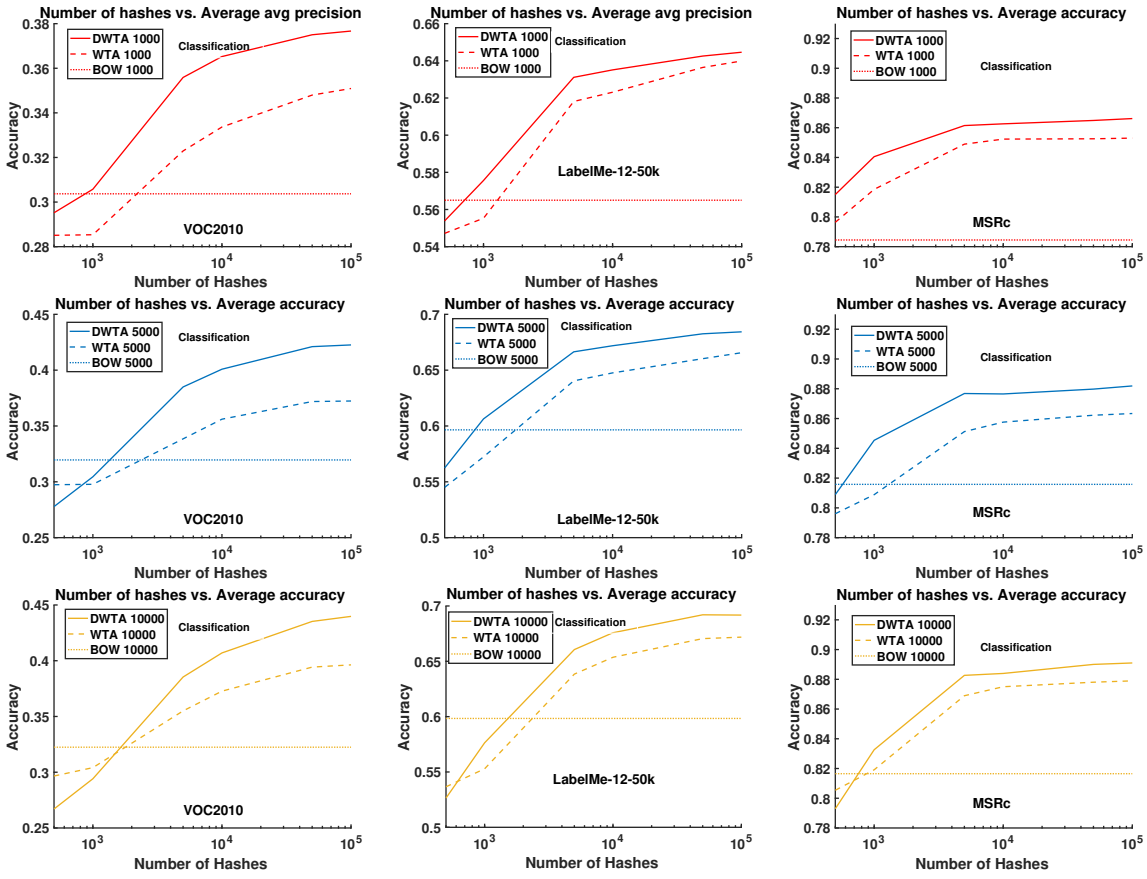

Figure 2: Densified WTA vs. WTA on the task of Image Classification on three different vision datasets. We used 1000, 5000 and 10000 BoW representation of the images. The y-axis is the mean accuracy and the x-axis is the number of hashes used as features. The horizontal lines (dotted) are classification based just on the BoW features. The semi-dotted lines are the vanilla WTA hashes and bold lines are our proposed Densified WTA Hashes. The colors represent which BoW was used as features. Densified WTA significantly outperforms the corresponding WTA consistently for all the choices.

For each query image, the nearest-neighbors of each test data were ranked among training data based on the Hamming distance of the hash codes. Since we had labeled datasets, all the images with the same label as the query were treated as the gold standard neighbors. Note, as mentioned in our proposal, WTA and Densified WTA leads to two different similarity measures (or kernel). Therefore, this experiment is comparing which among these two kernels agrees with the ground truth labels.

Replicating the setting of the original WTA paper, we first generated standard BoW of local descriptors, computed from the images, using the publicly available code [Vedaldi and Fulkerson, 2010]. BoW was generated by extracting local descriptors from the dense grid over each image and quantizing them using K-means. We used DSIFT [Kokkinos et al., 2012] as our descriptor measuring gradient at each key point pixel. The gradient was represented by a single 128-dimensional vector, stacked by a three-dimensional $(8 \times 4 \times 4)$ elementary feature vector formed by the pixel location (4×4) and the gradient orientation (8). In this experiment, we consider BoW with 1000, 5000, and 10,000 bins to demonstrate the effect of sparsity. We then generated WTA and Densified WTA hashes from these images and produce feature vectors as suggested in the WTA paper. For the feature generation, we used the fixed recommended setting of $K = 4$ for all the datasets which was picked using the same method described in [Yagnik et al., 2011] and best for WTA. The precision and recall curves for the rankings based on different hash codes are shown in Figure 1. We show plots for 64, 256 and 512 hash codes of 1000, 5000 and 10000 BoW representations (9 curves for each dataset per hashing scheme). To average out the randomness of both Densified and original WTA hashing, every curve on the graphs is averaged from 10 runs.

Densified WTA hashes lead to notably better precision-

recall compared to Vanilla WTA on all combinations irrespective of the choices of the dataset. As with classification, an increase in BoW leads to larger gap due to increases in sparsity. This again validates our claims. It is exciting to see that a small but principle modification to WTA Hashing can lead to drastic benefits.

5.3 CLASSIFICATION

Our motivation for comparing two Hashing algorithms using classification accuracy is to quantify the discriminative power in hashes. We use Densified WTA codes to do the classification task on the VOC2010, LabelMe-12-50k and MSRc datasets. We don't compare with those state-of-the-art methods like a particular type of nonlinear Mercer kernels, e.g. the intersection kernel or the Chi-square kernel [Yang et al., 2009] in classifying these datasets. Instead, we apply Densified WTA and original WTA hashes to a baseline method, sparse BoW of local descriptors and passing to linear SVM classifier, to show that the Densified WTA achieve superior improvement on classification tasks on sparse data.

To compute the classification performance we ran a simple SVM on BoW features, WTA hashed features, and Densified WTA hashed features. We varied the number of hash features over a range of values: $5 \times 10^2, 1 \times 10^3, 5 \times 10^3, 1 \times 10^4, 5 \times 10^4, 1 \times 10^5$. We again choose K=4. The C parameter of SVM was tuned using cross-validation, for every individual run, to ensure the best possible performance on every combination of the number of features and the hashing scheme. This ensures fairness of the comparisons.

Figure 2 compares the mean average precision of classification tasks using Densified WTA codes, WTA codes and basic sparse BoW on three datasets. The baseline, mean average precision for the three BoWs with different bins is shown by dashed straight lines. The mean average precision for WTA feature vectors is shown by dot-dashed curves and for Densified WTA feature vectors is shown by dot-dashed curves. We observe that as stated in WTA paper, precision increases when original BoW bin number increases or the number of codes increases with WTA beating BoW in each case. These observations are in line with the original WTA paper. We followed the experiment pipeline from the WTA paper, while generating BoW using standard package [Vedaldi and Fulkerson, 2010]. It is not surprising to see exactly the same trends in classification results with a difference in relative values.

The Densified WTA consistently outperforms Vanilla WTA significantly on all the three datasets, irrespective of the choice of BoW or the number of hashes. Moreover, the performance gap increases with the number of BoW. The increase in BoW rises sparsity of the dataset and hence this trend validates our hypothesis and theory in this paper. The gains over WTA are significant and our results clearly push the boundary of classification performance with hashing-based kernels significantly outperforming BoW. Note that increasing BoW from 5000 to 10000 leads to no gains in accuracy. But with hashing, especially Densified WTA, the gains keep climbing.

Table 6: Average running time comparison among DWTA, and Sparse Random Projection for three datasets with 10000 BoW representation and 512 hashes.

	SRP (ms)	**DWTA (ms)**
VOC2010	8.578	0.032
LabelMe-12-50k	8.62	0.046
MSRc	8.609	0.04

5.4 RUNNING TIME OF DWTA HASHING

As mentioned in Section 4.3, DWTA hashing only induces negligible cost of densification. We implemented DWTA and another popular algorithm for sparse data, Sparse Random Projection [Achlioptas, 2001] (SRP) and empirically show the average running time comparison of both algorithms for each data point in Table 6. We used 10000 BoW representation for three datasets and 512 hashes. The results clearly exhibited the advantage of DWTA over FRP in running time which further proves the superiority and Practicality of DWTA in general.

6 CONCLUSIONS

In this work, we revisited the problem of WTA Hashing for very sparse datasets which are ubiquitous in large-scale applications. We found a particular issue with WTA hashing in this regime which makes them uninformative with an increase in sparsity. We provide a principled solution to this problem using the novel 2-universal hashing for "Densification". Our solutions leverage the theoretical benefits of rank correlation methods and at the same time successfully resolves the concern of uninformative hash values produced by WTA Hashing for data with high sparsity. Evaluation results shown confirm the superior performance of Densified WTA Hashing on both image retrieval and classification task.

ACKNOWLEDGEMENTS

This work was supported by National Science Foundation IIS-1652131, RI-1718478, AFOSR-YIP FA9550-18-1-0152 and Amazon Research Award.

References

(2004). Microsoft research cambridge object recognition image database. Microsoft Research.

Achlioptas, D. (2001). Database-friendly random projections. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 274–281. ACM.

Caiafa, C. F., Sporns, O., Saykin, A., and Pestilli, F. (2017). Unified representation of tractography and diffusion-weighted mri data using sparse multidimensional arrays. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4340–4351. Curran Associates, Inc.

Canini, K., Chandra, T., Ie, E., McFadden, J., Goldman, K., Gunter, M., Harmsen, J., LeFevre, K., Lepikhin, D., Llinares, T., et al. (2012). Sibyl: A system for large scale supervised machine learning. *Technical Talk*.

Chapelle, O., Haffner, P., and Vapnik, V. N. (1999). Support vector machines for histogram-based image classification. *Neural Networks, IEEE Transactions on*, 10(5):1055–1064.

Chen, B., Shrivastava, A., and Steorts, R. C. (2017). Unique entity estimation with application to the syrian conflict. *arXiv preprint arXiv:1710.02690*.

Chen, B., Xu, Y., and Shrivastava, A. (2018). Lsh-sampling breaks the computational chicken-and-egg loop in adaptive stochastic gradient estimation.

Dean, T., Ruzon, M., Segal, M., Shlens, J., Vijayanarasimhan, S., and Yagnik, J. (2013). Fast, accurate detection of 100,000 object classes on a single machine. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1814–1821.

Everingham, M. and Winn, J. (2010). The pascal visual object classes challenge 2010 (voc2010) development kit.

Indyk, P. and Motwani, R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM.

Jiang, A., Bohossian, V., and Bruck, J. (2007). Floating codes for joint information storage in write asymmetric memories. In *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, pages 1166–1170. IEEE.

Kokkinos, I., Bronstein, M., and Yuille, A. (2012). Dense scale invariant descriptors for images and surfaces.

Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2006). Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808.

Li, P. and König, A. C. (2011). Theory and applications b-bit minwise hashing. *Commun. ACM*.

Li, P., Shrivastava, A., Moore, J. L., and König, A. C. (2011). Hashing algorithms for large-scale learning. In *Advances in neural information processing systems*, pages 2672–2680.

Liu, W., Shen, X., and Tsang, I. (2017). Sparse embedded k-means clustering. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 3319–3327. Curran Associates, Inc.

Liu, W. and Tsang, I. W. (2016). Sparse perceptron decision tree for millions of dimensions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 1881–1887. AAAI Press.

Liu, W. and Tsang, I. W. (2017). Making decision trees feasible in ultrahigh feature and label dimensions. *Journal of Machine Learning Research*, 18(81):1–36.

Parikh, D. and Grauman, K. (2011). Relative attributes. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 503–510. IEEE.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184.

Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173.

Shrivastava, A. (2015). *Probabilistic Hashing Techniques For Big Data*. PhD thesis, Cornell University.

Shrivastava, A. (2017). Optimal densification for fast and accurate minwise hashing. *arXiv preprint arXiv:1703.04664*.

Shrivastava, A. and Li, P. (2014a). Densifying one permutation hashing via rotation for fast near neighbor search. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 557–565.

Shrivastava, A. and Li, P. (2014b). Improved densification of one permutation hashing. In *UAI*, Quebec, CA.

Spring, R. and Shrivastava, A. (2017). Scalable and sustainable deep learning via randomized hashing. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 445–454. ACM.

Vedaldi, A. and Fulkerson, B. (2010). Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1469–1472. ACM.

Vijayanarasimhan, S., Shlens, J., Monga, R., and Yagnik, J. (2014). Deep networks with large output spaces. *arXiv preprint arXiv:1412.7479*.

Weber, R., Schek, H.-J., and Blott, S. (1998). A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, pages 194–205, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Yagnik, J., Strelow, D., Ross, D. A., and Lin, R.-s. (2011). The power of comparative reasoning. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2431–2438. IEEE.

Yang, J., Yu, K., Gong, Y., and Huang, T. (2009). Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801. IEEE.

Ziegler, A., Christiansen, E., Kriegman, D., and Belongie, S. (2012). Locally uniform comparison image descriptor. In *Neural Information Processing Systems (NIPS)*, pages 1–9, Lake Tahoe, NV.