Gaussian Normalization: Handling Burstiness in Visual Data

Remi Trichet and Noel E. O'Connor
Insight centre for data analytics
Dublin City University, Glasnevin, Ireland
remi.trichet@gmail.com, noel.oconnor@dcu.ie

Abstract

This paper addresses histogram burstiness, defined as the tendency of histograms to feature peaks out of proportion with their general distribution. After highlighting the impact of this growing issue on computer vision problems and the need to preserve the distribution information, we introduce a new normalization based on a Gaussian fit with a pre-defined variance for each datum that suppresses burst without adversely affecting the distribution. Experimental results on four public datasets show that our normalization scheme provides a staggering performance boost compared to other normalizations, even allowing Gaussian-normalized Bag-of-Words to perform similarly to intra-normalized Fisher vectors.

1. Introduction

Feature and feature space normalization are important steps in most multimedia applications [24, 11, 27]. When carefully tuned, this now de-facto pre-processing step can substantially boost the application performance. However, understanding which normalization may work best for a given application remains almost empirical. Recently [3, 10], *burstiness* as been identified has one influential factor in this direction.

Burstiness is defined as *"the property that a given visual element appears more times in a visual media than a statistically independent model would predict"* [9]. This phenomenon, first stated in text retrieval [3, 10], refers to the trend of codewords to appear in groups within a given visual media, leading to a representation biased toward them. Repetitive patterns of textures [14] and the increasing trend to use dense features have worsened its impact.

More practically, it characterizes the trend of histograms to feature peaks out of proportion with their general distribution. Bursty histograms will typically display a few bins with very high values. Codeword frequency is inherently linked to the Bag-of-Word (BoW) representation [13]. So, since visual vocabulary generation is the core component of any encoding technique, whether you are using a distribution [13] or residuals [8, 20] based encoding, these bins will strongly influence any comparison metric to therefore neglect the remaining information. This, of course, has a major impact on classifier decision.

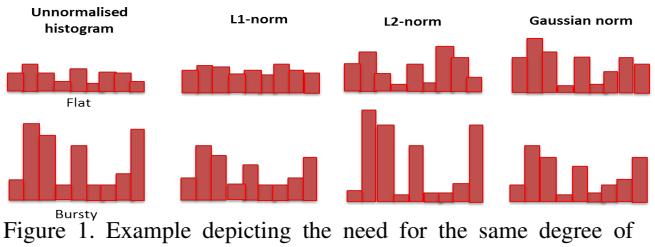

Figure 1. Example depicting the need for the same degree of burstiness in each data point. The same norm is applied to both, a flat and a bursty histogram. L^1-norm flattens both histograms whereas L^2-norm increases their peaks. Gaussian normalisation seeks the same optimal variance for each datum, therefore facilitating comparisons.

On the other hand, overly flat histograms will lack discriminative power. Hence, a certain magnitude of peaks, depending on the dataset and the encoding scheme, is required to optimize system performance.

One key idea of this article stems from the fact that methods tackling the issue generally model *burstiness* as an independent component of the data [3, 6, 10, 15, 23, 22]. However, these techniques often reach their limits in the context of complex or big data, typically encountered in computer vision problems. Therefore, in this case, the common practice is to simply normalize the data accordingly [1, 25, 7]. In both cases, only the *burstiness* of the overall dataset is considered.

Consequently, dataset *burstiness* is reduced or increased in relation to the needs. If this process improves the overall performance over a dataset, it may nevertheless have a harmful effect on some individual data samples, the peak magnitudes necessarily varying from one data point to another. For instance, lets assume that a normalization B, leading to flatter histograms, is more adapted than a normalization A for a given classification task. Normalization B will certainly harm the proportion of data that were already displaying rather flat histograms with norm A, making these data samples even more difficult to classify. Figure 1 illustrates this idea.

More generally, the optimal histogram peak magnitude required for computer vision applications has never been clearly studied. The recently proposed intra-

normalization [1] efficiently deals with the issue by performing component-wise normalization. However, this strategy is performed at the cost of the codeword distribution and is restricted to residual-based encoding strategies.

The core assumption of this paper is that this phenomenon is detrimental to the data representation. After showing its negative effect, we will introduce a new normalization scheme that limits its impact and prove its efficiency via extensive experimentation. Our new normalization scheme yields the same peak magnitude for every single data point histogram while preserving the distribution information. It is simple, handles sparse representations, and always improves performance compared to other normalizations. The method is grounded on the assumption that every single data point in the ordered distribution can be represented with the same Gaussian distribution. The normalization process unfolds as follows. Histogram bins are reordered to closely fit a Gaussian. Next, the bin values are modified to display a predefined variance σ_0^2. Finally, the bins are reassigned to their original position.

Our normalization scheme has the following advantages:

- It guaranties that every single data point will have the exact same peak magnitude, therefore avoiding bursty components.
- It preserve the distribution information.
- To the best of our knowledge, this is the first normalization process that efficiently deals with *burstiness* for both, Bag-of-Words and residual based encoding strategies.

The rest of this paper is organized as follows. Section 2 reviews the state of the art. Section 3 details our Gaussian based normalization. Extensive experimentation is presented in section 4. Conclusion and future work are discussed in the final section.

2. Related Work

Few approaches have been proposed to specifically tackle *burstiness*. Existing strategies can broadly be divided in two types. The first one attempts to model *burstiness* as a noisy component that can subsequently be discarded. Poisson distribution [3], K-mixtures [10], and LDA [4] models have been proposed and successfully applied in the context of text retrieval [6], text clustering [15], and spatio-temporal term mining [28]. [26] utilized feature self-similarity to discard similar features within images. [23, 22] introduced a penalty weighting scheme to attenuate the visual word count differences as their raw value grows. However this type of model cannot reduce the model variance without increasing its bias. In other words it fails to differentiate feature bursts from the genuine peaks and displays a limited

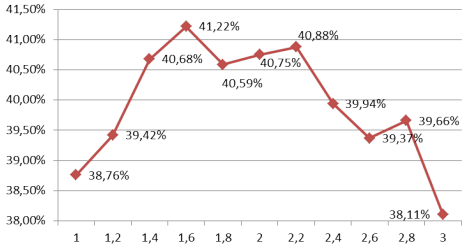

Figure 2. Peak magnitude influence on mean average precision performance. Tests are run on the HOHA dataset, varying the parameter a of the L^a-norm.

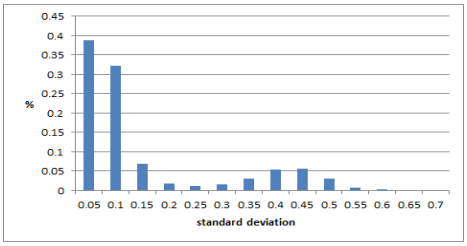

Figure 3. Distribution of histogram standard deviations within the HOHA dataset.

efficiency as the amount of noise and outliers increases.

Consequently, computer vision applications, which are significantly prone to these kinds of difficulties, often prefer to use an apt normalization scheme to limit the impact of bursty values. L^1 and L^2 norms [7] tests are the typical choices. More recently, with the surge of feature encoding methods [2], Intra- [1] and power normalization [25, 21] emerged with VLAD [8] and Fisher vector encoding [20] methods. This type of feature representation is based on residuals, or feature value average difference with their closest codeword. As such, the overall distribution bears less importance. By square-rooting each bin, the power-normalization [25] reduces their impact while keeping the distribution information intact. By performing component-wise normalization, intra-normalization [1] successfully deals with the *burstiness* but at the cost of the distribution information. [27] chose to detect and suppress bursts early for faster representation. To sum up, all these approaches improve the performance but always dealing with the burstiness to a limited extent or at the cost of some information (like the distribution), therefore leaving space for improvement.

3. Gaussian Normalization

The main hypotheses underpinning our work is that (1) more important than counterbalancing histogram *burstiness* impact on performance, a precise peak magnitude must be set for optimal performance, and (2) that, peak magnitude varying for each data point, each histogram must be normalized independently. The first assumption can easily be verified with a simple experiment. Figure 2 plots the event

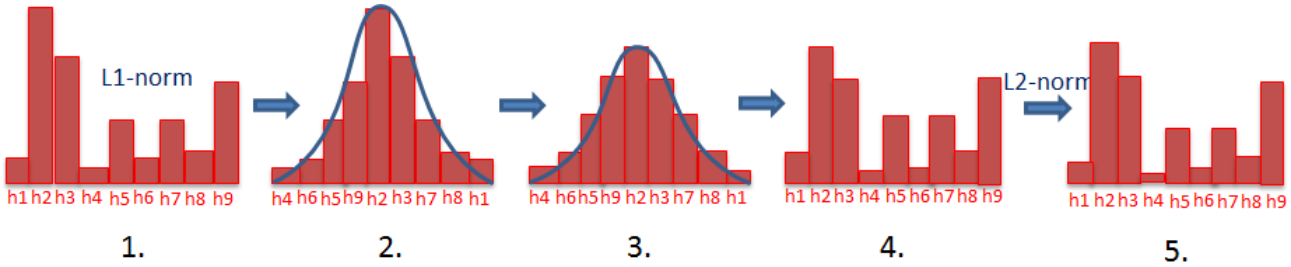

Figure 4. Gaussian normalization process. 1- (optional) L^1-normalized histogram. 2-Re-ordering the bins to fit a Gaussian shape. 3-Modifying the values so that each bin will have the value of a Gaussian with predefined variance σ_0^2 4- Re-ordering the bins to match their original position in the histogram. 5- L^2-normalization to handle data sparsity.

recognition performance on the HOHA dataset [12], while varying the a value of L^a normalized histograms encoded using a dictionary of 4K Bag-of-Words. By changing a, we directly tune the histogram peak magnitude. It confirms that low peak magnitude (i.e. L^1-norm) as well as bursty ones (i.e. L^3-norm) both lead to sub-optimal performance. The best score is obtained for $a = 1.6$, close to the L^2-norm, commonly utilized with BoW. To validate the second assumption, we have plotted the distribution of histogram standard deviations within the HOHA dataset [12]. Results, presented in figure 3 show the discrepancy. 29.2% of the data have a standard deviation different from the bulk of the dataset, which is 0.075. Any normalization that aims to optimize the dataset histograms altogether will harm the histograms of this subset.

The main idea of this paper is that this damaged data representation will harm the final results. So, we introduce a new normalization that prevents this shortcoming by optimizing each histogram *burstiness* independently.

Our Gaussian normalization process, assumes that any data point distribution should match a normal distribution. Lets define the *burstiness B(H)* of a histogram *H* as its variance $\sigma^2(H)$:

$$B(H) = \sigma^2(H) = \frac{1}{n}\sum_{i=0}^{n}(h_i - \mu)^2 \tag{1}$$

Probing further, we aim for any histogram undertaking our Gaussian normalization process to have the same *burstiness* σ_0^2. Therefore, our normalization *N(.)* over a histogram *H* of *n* bins $[h_1 \dots h_n]$ should have the following 3 properties:

- It sums to 1.
$$\sum_{i=0}^{n} N(h_i) = 1 \tag{2}$$
- It preserves the bin value ordering.
$$N(h_i) < N(h_j) \Leftrightarrow h_i < h_j \quad \forall h_i, h_j \in H \tag{3}$$
- It endows all histograms with the same *burstiness*.
$$B(N(H1)) = B(N(H2)) = \sigma_0^2 \forall H1, H2 \tag{4}$$

As we cannot expect a given distribution of histogram *H* to closely match a normal distribution, we deal with its re-ordering *O(H)* in descending order:

$$O(h_i) = j | j_1 < j_2 \Leftrightarrow h_{i1} > h_{i2} \quad \forall O(h_{i1}) = j_1, O(h_{i2}) = j_2$$
$$O(h_i) \in [0 \dots n-1] \quad with \quad i \in [1 \dots n] \tag{5}$$

with *n* the number of bins and $O(h_i)$ the bijective function outputting the position of a bin i, with value h_i, after re-ordering histogram *H* . We want to fit this distribution to a normal distribution of mean 0 and variance σ_0^2:

$$N(h_i) = \frac{1}{\sigma_0\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_i}{\sigma_0}\right)^2} \tag{6}$$

In our case, we have $x_i = h_i/n$. Moreover, since we are dealing with a reordered distribution in descending order, we only consider one half of the Gaussian. Hence, the best fits to a Gaussian formalizes as:

$$N(h_i) = \frac{1}{\sigma_0\sqrt{2\pi}} e^{-\frac{1}{4}\left(\frac{O(h_i)}{n\sigma_0}\right)^2} \tag{7}$$

To make this normalization independent of the histogram size, the bin indexes h_i, and therefore their re-ordereing $O(h_i)$, are rescaled to fit the same range.

In practice, since the same n values $N(h_i)$ will be assigned to all histograms, those are pre-computed to save on the costly exponential calculations. Therefore, the only computation involved in the process is the sorting of the values.

For a better understanding, the normalization can be broken down into the following five steps, illustrated in figure 4: Taking as input a pre-normalized histogram to insure their variance will be learned on the same basis (Fig. 4.1.), we first re-arrange the bins to fit a Gaussian distribution (Fig. 4.2.). This also indirectly fits a Gaussian to it. Then, the bin values are modified to match a Gaussian function of predefined variance σ_0^2 with the same mean (Fig. 4.3.). Note that we are not rescaling the values according to this updated variance σ_0^2, as this would only limit the influence of bursty or flat values. Instead we actually assign the Gaussian function value to the bin. Therefore, at this point, any

histograms should have the same values, but distinct bin orderings. Fourth, we order the bins back to their original positions (Fig. 4.4.).

3.1. Handling sparsity

This normalization scheme wouldn't be efficient without a mechanism to handle data sparsity. Indeed, in this case, Gaussian value assignment (in phase 3.) leads to information loss by artificially and unnecessarily increasing the near-zero values.

To limit the information loss, it is, in theory, possible to extend this single Gaussian fitting scheme to a mixture of Gaussians, utilizing a maximum-a-posteriori parameter estimation method (like expectation-maximization [17] for instance). This strategy would lead to a closer fit to the data. However the bin value modification phase will lead to the violation of property (3), if based on more than one Gaussian. Consequently, we discontinued our investigations in this direction.

An alternate way to deal with information loss is to leave the small values untouched. Indeed, these values mostly represent noisy codeword presence within the analysed media and the purpose of this normalization is mainly to bring the bursty ones to a reasonable level. Therefore, our normalization does not modify values inferior to ϵ. This last step gives more flexibility to our Gaussian normalization, allowing it to preserve sparsity and deal with distributions similar to Dirichlet ones. However, our normalization violates property (2). Therefore, the final histogram is L^2-normalized to cope with this issue.

The only known failure case is the uniform distribution. However, such distributions already lead to bad performance and are highly unlikely in machine learning tasks.

4. Experiments

We indirectly measure the benefits of our burstiness handling strategy by using the Gaussian normalization for an event classification task. This section is divided in 5 subsections that respectively detail our datasets, testbed, results, analyses on the influence of the main parameter σ_0^2, and a discussion on possible variations.

4.1. Datasets

We use 4 different datasets of various complexities to assess the Gaussian normalization with 2 different multimedia applications: Event and object recognition.

The YouTube dataset [14] contains 1600 sequences divided in 11 action categories. We follow the original setup [14] using leave-one-out cross validation for a pre-defined set of 25 folds.

The Hollywood2 dataset [16] has been collected from 69 different Hollywood movies and includes 12 action classes. It contains 1,707 videos split into a training set (823 videos) and a test set (884 videos), coming from different movies. It is a challenging dataset, featuring unrelated footage, significant background, intra- and inter-classes variations, and camera motion.

The HOHA dataset [12] considers 8 human actions gathered from 32 movies. The dataset is divided into test and train sets, respectively containing 211 and 219 video clips from different movies. Even though they are different, the clips have been gathered from the same videos as the Hollywood2 dataset. So, it can roughly be considered as a subset of it. However, this benchmark is more challenging than hollywood2. We essentially tested on this dataset to study how our normalization responds to dataset scale changes.

The PASCAL VOC 2007 object recognition dataset [5] contains about 10000 images split into train, validation, and test sets, and labelled with 20 object classes. Significant noise, small objects, intra category variation, and inter-category similarities (ex: motorcycle and bicycle) make this dataset a challenge.

A one-vs-all SVM classifier is learned and evaluated independently for each category. Average accuracy across all classes is reported as performance measure for the Youtube dataset, and mean Average Precision (mAP) for other datasets.

4.2. Experimental setup

For these experiments, feature and encoding techniques were selected to allow the comparison with the most common normalization schemes: Intra-normalization [1], power normalization [25, 21] with Fisher vector coding, as well as typical L^2 normalization with Bag-of-Words (BoW) [13].

Videos are first rescaled to a 640 $\times$ 480 resolution, as applicable. We then employed DT features [29]. PASCAL 2007 images used with PHOW [18] features, following [2] setting. Features are encoded utilizing Bag-of-Words (BoW) [13] or Fisher vector (FV) coding [20]. BoW is based on k-means clustering with hard-assignment. The codebook size is 4K or 10K, determined over 500K randomly sampled feature vectors. The final histograms are determined using ANN [19] and then normalized. SVMs with χ^2 kernels are further employed for BoW. Our normalization scheme is compared with the L^2-norm under this setting.

We encode FV based on a mixture of 256 Gaussians. Linear SVMs are further utilized for all these runs. We experimented with Gaussian normalization solely and Gaussian normalization before intra-normalization. We compared with power- and intra-normalization. A final L^2-normalization is performed in every case.

Table 1. Comparison of our normalization method with the state-of-the art. The best value of σ_0^2 is in bold. We report average accuracy over all classes for the Youtube dataset, mean average precision over all classes for other datasets. **BoW** - Bag of Words. **FV** - Fisher vector encoding. **G** - Gaussian normalization. **I** - Intra normalization. **P** - Power normalization. $\boldsymbol{L^2}$- L^2-normalization.

Encoding type	codebook size	Norm	HOHA	Holywood2	Youtube	PASCAL2007
BoW	4K	L^2	40.82%	58.3% [29]	87.3%	53.42% [2]
	4K	G(σ_0^2)+L^2	44.12% (**0.8**)	61.51% (**0.8**)	91.61%(**0.4**)	56.12%(**0.5**)
	10K	L^2	41.77%	59.69%	87.65%	54.98% [2]
	10K	G(σ_0^2)+L^2	44.86% (**0.1**)	62.22% (**0.6**)	91.01%(**0.2**)	57.33%(**0.5**)
FV	256	I+L^2	45.39%	61.45%	90.89%	61.79%
	256	P+L^2	44.51%	60.1% [30]	90.35%	61.69% [2]
	256	G(σ_0^2) + L^2	46.03% (**0.9**)	65.14% (**1.5**)	92.82%(**1.5**)	63.85%(**1.4**)
	256	G(σ_0^2)+I+L^2	46.57% (**0.5**)	65.45% (**1.6**)	93.2%(**1.4**)	64.06%(**1.4**)

4.3. Results

Results are presented in table 1. Note that our baseline for the Youtube dataset (with a 4K codebook) is 3.1% higher than the one provided in [29]. We assume this difference is due to the recent dataset update. Also, we assume that similar Fisher vector base results on the PASCAL 2007 challenge are due to the original PCA performed on the PHOW features by authors in [2].

Results on the HOHA dataset show a considerable performance boost for our normalization scheme compared to the L^2-norm. Performance increases from 40.82% to 44.12% with a 4K codebook and from 41.77% to 44.86% with a 10K codebook. Applied jointly with Fisher vector encoding, Gaussian normalization and Gaussian intra-normalization respectively score 45.83% and 46.57%, that is 1.32% and 1.18% above power- and intra- normalization.

Similar improvement is stated on the Hollywood2 dataset. Performance increases from 58.3% to 61.51% with a 4K codebook and from 59.69% to 62.22% with a 10K codebook. These results are higher than intra-normalized Fisher vector, scoring 60.1% on this dataset. Applied jointly with Fisher vector encoding, Gaussian normalization scores 65.45%, that is 4% more than intra-normalization.

Performance on the Youtube dataset increases from 87.3% to 91.61% with a 4K codebook and from 87.65% to 91.01% with a 10K codebook. These results are also higher than intra-normalized Fisher vector, scoring 90.89% on this dataset. Applied jointly with Fisher vector encoding, Gaussian normalization and Gaussian intra-normalization respectively score 92.82% and 93.2%, which represents a 2.47% and 2.13% increment over power- and intra- normalization.

Finally, performance on the PASCAL 2007 challenge increases from 53.42% to 56.27% with a 4K codebook and from 54.98% to 57.33% with a 10K codebook. Applied jointly with Fisher vector encoding, Gaussian normalization scores 64.06%, which is 2.33% more than intra-normalization.

Overall, the Gaussian normalization leads to a considerable improvement, when combined with the Bag-of word model, allowing similar performance to intra-normalized Fisher vectors. While still producing the best results our normalization scheme leads to a more modest boost when combined with Fisher vector encoding, with the exception of the Hollywood2 dataset.

Five conclusions can be drawn from these numbers. First, this work shows that keeping the histogram representation under the control of a predefined distribution model is paramount for better classification performance. Despite the obvious success of the Gaussian based model for this purpose, further work may be required to assess if other distribution models, like the Weibull distribution for instance, may be a better fit.

Second, applying this normalization to residual based encoding strategies leads to a significant improvement, though, on average, moderate compared to distribution-based encoding techniques. This makes sense as the competing normalizations share some properties with the Gaussian normalization: Power normalization preserves the original distribution, and, by normalizing each component independently, intra-normalization also keeps each histogram *burstiness* similar. However, and in contrast to Gaussian normalization, intra-normalization discards the codeword distribution information, therefore resulting in slightly lower performance.

Third, applying Gaussian normalization alone performs favourably, always resulting in better performance than intra-normalization. This further emphasizes the importance of the distribution when properly handled.

Fourth, intra-normalization remains useful. Used after Gaussian-normalization, it still brings an additional 0.31% to 1.44% performance increment.

Finally and more importantly, intra-normalization of Fisher vectors and Gaussian normalization of Bag-of-Words histograms lead to similar performance. This calls into question the predominance of residual-based representation over the codeword distribution one. These two strategies may be complementary and their fusion may lead to further improvement.

4.4. Parameter influence

The main parameter influencing the normalization is of course σ_0^2. We ran extensive experiments to evaluate the extend of the affect of this parameter. Results are presented in figures 5 to 8.

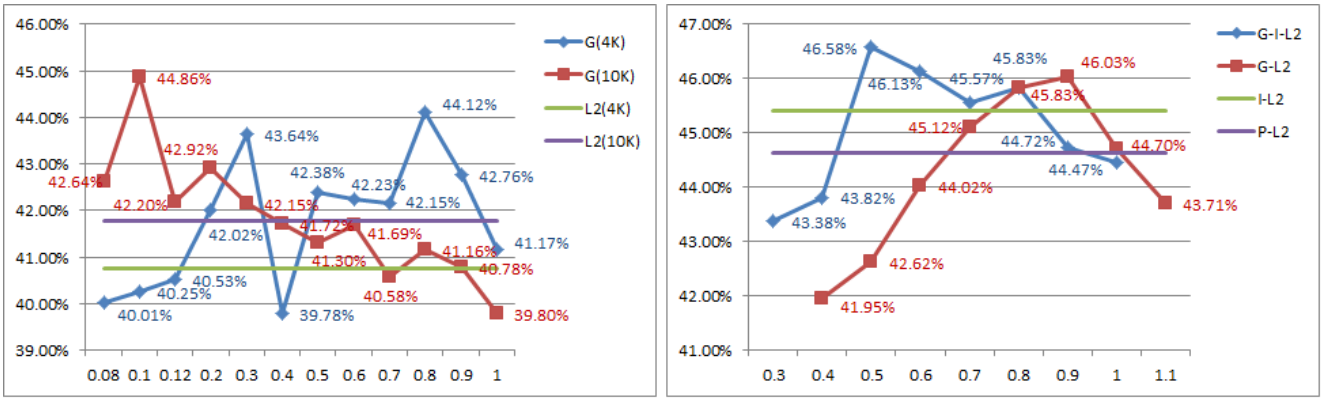

Figure 5. Mean average precision results on the HOHA dataset in relation to σ_0^2 for Bag-of-Words strategy using 4K and 10K codebook sizes (*left*) and for Fisher vector encoding (*right*). Best viewed in colour. **G** - Gaussian normalization. **I** - Intra normalization. **P** - Power normalization. **L2** - L^2-normalization.

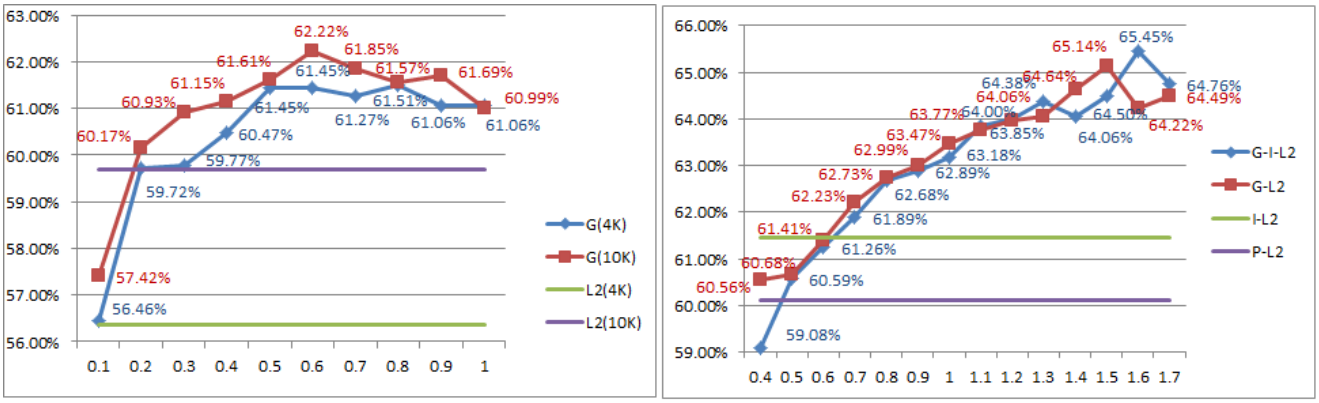

Figure 6. Mean average precision results on the Hollywood2 dataset in relation to σ_0^2. for Bag-of-Words strategy using 4K and 10K codebook sizes (*left*) and for Fisher vector encoding (*right*). Best viewed in colour. **G** - Gaussian normalization. **I** - Intra normalization. **P** - Power normalization. **L2** - L^2-normalization.

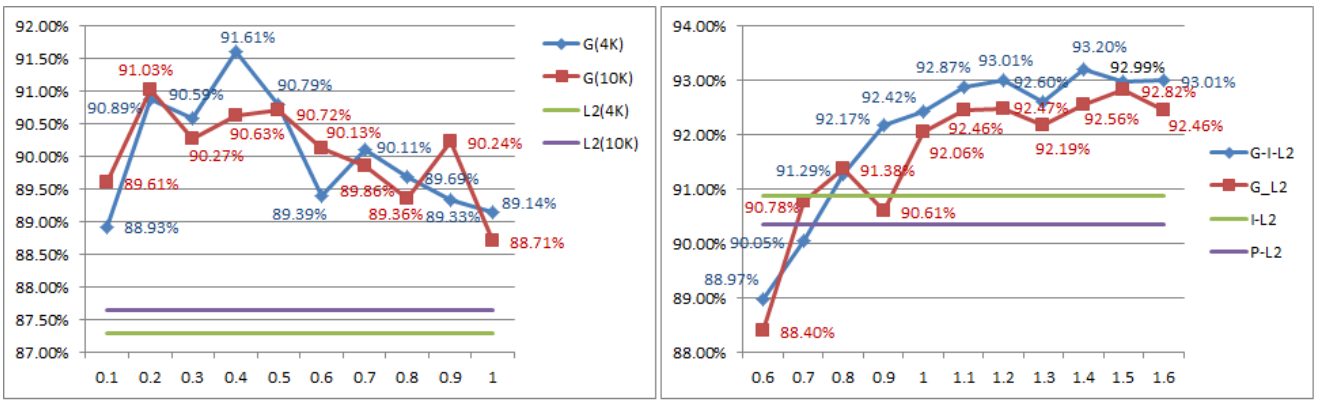

Figure 7. Average accuracy results on the Youtube dataset in relation to σ_0^2. for Bag-of-Words strategy using 4K and 10K codebook sizes (*left*) and for Fisher vector encoding (*right*). Best viewed in colour. **G** - Gaussian normalization. **I** - Intra normalization. **P** - Power normalization. **L2** - L^2-normalization.

Overall, the smooth curves show the reliability of the normalization. Applied to both, BoW and Fisher vector encodings, the Gaussian normalization performs better than its competitors for a wide range of σ_0^2 values on the Hollywood2 and Youtube datasets. It scores similarly or better for most values of σ_0^2 on the HOHA dataset. This further demonstrates the effectiveness of the method.

As the curves show a global maxima that differs from one dataset to another, parameter estimation is required for optimal classification results. As shown in table 2, cross-

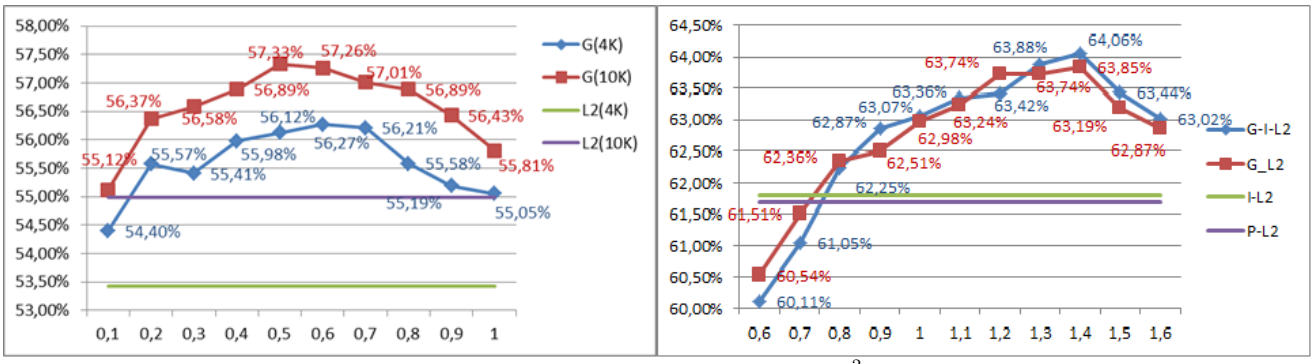

Figure 8. Mean average precision results on the PASCAL 2007 dataset in relation to σ_0^2. for Bag-of-Words strategy using 4K and 10K codebook sizes (*left*) and for Fisher vector encoding (*right*). Best viewed in colour. **G** - Gaussian normalization. **I** - Intra normalization. **P** - Power normalization. **L2** - L^2-normalization.

Table 2. Gaussian normalization results with estimation of σ_0^2 through cross validation. The estimated value of σ_0^2 is in bold. We report average accuracy over all classes for the Youtube dataset, and mean average precision over all classes for other datasets. **BoW** - Bag of Words. **FV** - Fisher vector encoding. **G** - Gaussian normalization. **I** - Intra normalization. L^2 - L^2-normalization.

Encoding type	codebook size	Norm	HOHA	Holywood2	Youtube	PASCAL2007
BoW	4K	G(σ_0^2)+L^2	42.87% (**0.73**)	61.45% (**0.54**)	90.99%(**0.34**)	55.87%(**0.52**)
	10K	G(σ_0^2)+L^2	43.75% (**0.09**)	61.98% (**0.56**)	90.51%(**0.37**)	56.42%(**0.48**)
FV	256	G(σ_0^2) + L^2	45.85% (**0.81**)	64.84% (**1.44**)	92.56%(**1.4**)	63.58%(**1.32**)
	256	G(σ_0^2)+I+L^2	45.96% (**0.63**)	64.12% (**1.41**)	92.96%(**1.36**)	63.77%(**1.36**)

validation can reliably be used for this purpose. The average performance drop is 0.53%, with a slight trend to underestimate σ_0^2 optimal value. A strategy to limit the performance loss in this case could be to segment the dataset in k clusters, learn a separate σ_0^2 for each one of them, and apply the appropriate one at testing time.

Based on these experiments, our findings concerning this optimal value of σ_0^2 are the following:

- Sparsity significantly impacts σ_0^2. It can be linked to two factors. First the codebook size. Indeed, more codewords means more histogram bins with low values. Sparser histograms having a lower variance, the optimal value of σ_0^2 will decrease as the codebook size increases. Second, the video clip sizes. Short clips often having a smaller raw feature count. For example, the HOHA dataset has rather short clips and sees the optimal value of σ_0^2 plummeting as the codebook increases from 4k to 10K.
- Fisher vectors require a higher value of σ_0^2. Typically $\sigma_0^2 > 1$. It is sensible as strong peaks usually deserve residual based encodings that also have few values close to zero. This leads to a flatter distribution than BoW encodings.
- Preliminary experiments did not reveal any correlation between the average variance of a dataset's histograms (before normalization) and its optimal value σ_0^2.
- Moreover, The optimal value of σ_0^2 does not seem to be only dataset dependent but also category dependent. Indeed, if the optimal mAP grows steadily toward the optimum value, there is a significant discrepancy of the per category best value for σ_0^2. Therefore automatic parameter estimation may also lead to a gain in performance.

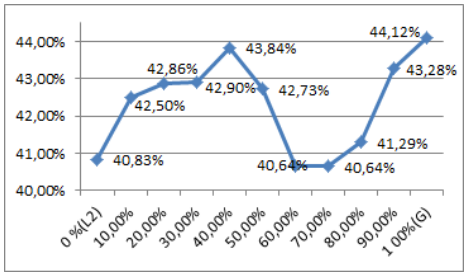

Figure 9. Mean average precision results on the HOHA dataset of the fusion of L^2- and Gaussian-normalized histograms with $\sigma_0^2 = 0.8$. Histograms are encoded with the Bag-of-Words strategy. The horizontal axis represents the Gaussian normalization fusion percentage from 0 (i.e. $L^2 - norm$) to 1 (i.e. Gaussian-norm).

- Finally, σ_0^2 or the normalization performance are not dataset size, nor application dependent. Indeed, HOHA and Hollywood have video clips extracted from the same original movies but share the same optimal variance σ_0^2 for a 4K codebook and display approximately the same performance boost. Similarly, object and event recognition applications share the same performance boost and parameter values.

4.5. Possible variations

This subsection discusses two straightforward variations on our normalization scheme. It aims at discarding invalid future research directions.

First, an alternate way to deal with information loss is to interpolate the bin values of the Gaussian normalization with a typical L^2 one. Experimentation and analysis of this variant on quantized histograms of codewords for the HOHA dataset is provided in figure 9 with $\sigma_0^2 = 0.8$ and shows lower performance compared to Gaussian-normalization solely. We assume this is due to the big difference between the two norm independent performance.

Second, the combination of Gaussian- and intra-normalization, aiming to take the best of these two strategies, can also be considered. But, applying it after intra-normalization does not increase the results. This makes sense as the intra-normalization destroys the distribution information and therefore the benefits of the Gaussian normalization.

5. Conclusion

In this paper, we presented a new normalization scheme dealing with histogram *burstiness*. Our normalization, harnessing a Gaussian fit on reordered histogram bins, is easy to implement and leads to considerable performance boost. Our future work first priority is to find a way to automatically calibrate σ_0^2. Fusion of residual and distribution based histograms also looks like a promising research direction.

6. Acknowledgment

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under grant number SFI/12/RC/2289.

References

[1] R. Arandjelovic and A. Zisserman. All about vlad. *CVPR*, 2013.

[2] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. *BMVC*, 2011.

[3] K. W. Church and W. A. Gale. Poisson mixtures. *Natural Language Engineering*, (1):163–190, 1995.

[4] G. Doyle and C. Elkan. Accounting for burstiness in topic models. *ICML*, 26:281–288, 2009.

[5] M. Everingham, A. Zisserman, C. Williams, and L. V. Gool. The pascal visual obiect classes challenge 2007 (voc2007) results. *Technical report, Pascal Challenge*, 2007.

[6] Q. He, K. Chang, and E.-P. Lim. Using burstiness to improve clustering of topics in news streams. *IEEE International Conference on Data Mining*, 2007.

[7] R. A. Horn and C. R. Johnson. Norms for vectors and matrices. *Ch. 5 in Matrix Analysis. Cambridge, England: Cambridge University Press*, 1990.

[8] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. *CVPR*, 2010.

[9] H. Jgou, M. Douze, and C. Schmid. On the burstiness of visual elements. *IEEE Computer society*, 2009.

[10] S. M. Katz. Distribution of content words and phrases in text and language modeling. *Natural Language Engineering*, (2):15–59, 1996.

[11] D. Kotsakos, T. Lappas, D. Kotzias, D. Gunopulos, N. Kanhabua, and K. Nrvg. A burstiness-aware approach for document dating. *SIGIR*, 2014.

[12] I. Laptev, M. Marszaek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. *CVPR*, 2008.

[13] D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. *ECML*, pages 4–15, 1998.

[14] J. Liu, J. Luo, and M. Shah. Recognizing realistic actions from videos in the wild. *CVPR*, 2009.

[15] R. E. Madsen, D. Kauchak, , and C. Elkan. Modeling word burstiness using the dirichlet distribution. *ICML*, 2005.

[16] M. Marszaek, I. Laptev, and C. Schmid. Actions in context. *CVPR*, 2009.

[17] G. McLachlan and D. Peel. Finite mixture models. *Wiley*, 2000.

[18] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI*, 27(10):1615–1630, 2005.

[19] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithmic configuration. *VISAPP*, 2009.

[20] F. Perronnin and C. Dance. Fisher kenrels on visual vocabularies for image categorizaton. *CVPR*, 2006.

[21] F. Perronnin, Y. Liu, J. Sanchez, , and H. Poirier. Large-scale image retrieval with compressed fisher vectors. *CVPR*, 2010.

[22] D. Qin, Y. Chen, M. Guillaumin, and L. V. Gool. Learning to rank bag-of-word histograms for large-scale object retrieval. *BMVC*, 2014.

[23] J. Revaud, M. Douze, and C. Schmid. Correlation-based burstiness for logo retrieval. *ACM multimedia*, pages 965–968, 2012.

[24] S. Ross, P. Mineiro, and J. Langford. Normalized online learning. *UAI*, 2013.

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

[26] F. Schaffalitzky and A. Zisserman. Automated location matching in movies. *Computer Vision and Image Understanding*, (92):236–264, 2003.

[27] M. Shi, Y. Avrithis, and H. Jegou. Early burst detection for memory-efficient image retrieval. *CVPR*, 2015.

[28] T.Lappas, M. R. Vieira, D. Gunopulos, and V. J. Tsotras. On the spatiotemporal burstiness of terms. *ACM multimedia*, 2012.

[29] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. *CVPR*, 2011.

[30] H. Wang and C. Schmid. Action recognition with improved trajectories. *ICCV*, 2013.