ACOUSTIC SCENE CLASSIFICATION USING DISCRETE RANDOM HASHING FOR LAPLACIAN KERNEL MACHINES

Abelino Jiménez, Benjamín Elizalde [†] *and Bhiksha Raj*

Carnegie Mellon University, Department of Electrical and Computer Engineering, Pittsburgh, PA, USA
abjimenez@cmu.edu, bmartin1@andrew.cmu.edu, bhiksha@cs.cmu.edu

ABSTRACT

State of the art acoustic scene classification techniques often employ features of large dimensionality, which are then used to train and perform inferences with kernel machines such as Support Vector Machines. However, the complexity of computing the non-linear kernel matrix for these methods increases with the dimensionality of the features and the size of the dataset. In this work, we introduce a new scheme that hashes features, which combined with a linear function approximates a non-linear Laplacian kernel. Each hash typically has lower dimensionality than the input features and each component is represented by one bit instead of floating values. Hence, allowing efficient computation of the kernel matrix using XOR operations rather than dot-products. Our scheme is demonstrated mathematically and tested in the 2017 DCASE: Acoustic Scene Classification. The hashes reduce up to six powers of two the feature representation with minimal loss of accuracy.

Index Terms— Acoustic Scene Classification, Laplacian Kernel, Kernel Machines, Random Features, Hash

1. INTRODUCTION AND RELATED WORK

Acoustic Scene Classification (ASC) aims to identify a recording as belonging to a predefined set of scene classes that characterizes an environment, for example *park*, *home*, or *office*. Typically, ASC approaches capture the diverse characteristics from the audio signal by computing different types of features, either hand-crafted [1, 2, 3, 4, 5] or derived from Neural Networks [6, 7, 8]. These features are commonly of high-dimensionality (in the order of thousands) and some state of the art approaches used them to train and make inferences with Support Vector Machines (SVMs), the best-known member of kernel machines.

Through appropriate choice of kernel SVMs can approximate non-linear functions or decision boundaries given enough training samples, however, the computation of the kernel matrix scales poorly. The Kernel matrix's time and storage complexity increases in the dimensionality and number of the inputs, which contrasts to linear functions, which can be computed much faster [9]. One solution to take the best of both approaches is by computing random features [10, 11, 12], which maps the input features into a randomized lower-dimensional feature space. Then, the resulting random features are combined in a linear manner to approximate a non-linear kernel, but at a much lower cost than direct computations of kernels.

Although kernel methods are common in ASC, random features are unexplored in this topic. We applied random features to ASC in [13] and reduced the dimensionality of the input features by one third with minimal loss of performance. Further improvements can be achieved by quantizing the random features.

One effective method for quantizing features is through *hashing* schemes [14]. Hashing schemes, roughly speaking, quantize the random features to be represented with just one bit instead of real numbers represented by 32 or 64 bits. This has significant implication in storage, because even if the dimensionality of the random features is the same or larger than the input features dimension, the overall size in bits of the transformed vector can be an order of magnitude smaller. This reduction has benefits in storage, processing and transmission [14].

Random features and hashing schemes are kernel dependent; this means that to approximate different non-linear kernels we need different kinds of random transformations of the input features. The most popular and best-studied kernels are shift-invariant [15, 16], such as Gaussian, Laplacian and Cauchy. The Gaussian employs the ℓ_2 norm and the Laplacian employs the ℓ_1 norm, and although both offer different benefits, the latter has been significantly less studied. The Laplacian kernel, in particular, has outperformed the Gaussian kernel without hashing schemes for sound-event classification [17, 18] and with hashing schemes for image processing [19, 20]. However, to the best of our knowledge, the mathematical and theoretical guarantees to employ hashing schemes to approximate Laplacian kernels was unavailable in the literature.

In this paper, we introduce a new scheme that hashes features, which are combined with a linear function to approximate a non-linear Laplacian kernel. Additionally, the hashing reduces the dimensionality of real-valued vectors into bits-based representations by reducing storage up to *six powers of two*. We evaluated our hashing scheme in the context of the 2017 DCASE Task 1 - Acoustic Scene Classification [21]. Our main contribution are the mathematical and theoretical guarantees to validate the approximation of the Laplacian kernel, which was unavailable in the literature and can be applied to tasks other than audio.

The organization of the paper is as follows: Section 2 describes our hashing scheme and shows theoretical guarantees to support how it approximates the Laplacian Kernel. Additionally, we explain how the training and inference steps applied to an SVM can be simplified using our approach. In Section 3, we describe our experiments and results on Acoustic Scene Classification, which are discussed in Section 4. Finally, in Section 5 we summarized our work.

2. METHODS

In this section we describe the construction of our hashing scheme and provide theoretical guarantees to validate its approximation to the Laplacian kernel, which was unavailable in the literature. In addition, we describe its application to Support Vector Machines. The full proofs of the following theorems are in our archive [1].

[†] Acknowledges CONACYT for his doctoral fellowship, No.343964

[1] http://www.andrew.cmu.edu/user/abelinoj/icassp2018/laplacian/app.pdf

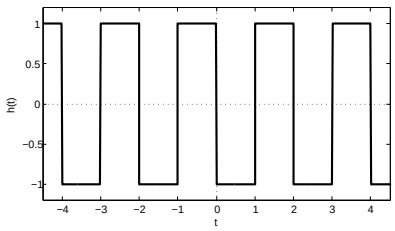
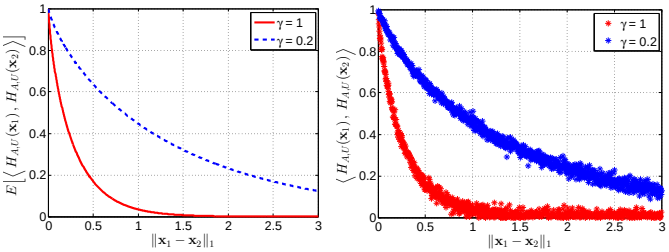


Fig. 1. The function h quantizes real values to -1 or 1 for each hash component.



(a) Theoretical Expression (b) Simulations

Fig. 2. **(a) Theoretical Expression.** Expectation of the inner product between $H_{A,U}(\mathbf{x}_1)$ and $H_{A,U}(\mathbf{x}_2)$ as a function of $\|\mathbf{x}_1 - \mathbf{x}_2\|_1$ using Theorem 2 considering $\gamma = 1$ and $\gamma = 0.2$. We took the first 5,000 terms of the series. **(b) Simulations.** Inner product between $H_{A,U}(\mathbf{x}_1)$ and $H_{A,U}(\mathbf{x}_2)$ as a function of $\|\mathbf{x}_1 - \mathbf{x}_2\|_1$ with $N = 5000$, $M = 2500$ and $\gamma = 1$ and $\gamma = 0.2$.

2.1. Hashing Scheme to Approximate Laplacian Kernel

In this subsection we show the definition of our hashing function, how the inner product between hashes is related to the Laplacian kernel and bounds of the expected value of such inner products.

The hashing scheme consists of a random transformation based on the Cauchy distribution, which is fundamental for our proposal. This distribution belongs to the p-stable distribution family, which means that any linear combination of independent random variables distributed as Cauchy will also be distributed as Cauchy. We use this property to prove the main theorem of this paper. In this paper, we considered the following definition.

Definition. Let X be a continuous random variable and $\gamma > 0$. Then, X is distributed as Cauchy(γ) if its density function is given by:

$$f(x\,;\,\gamma) = \frac{1}{\pi} \cdot \frac{\gamma}{x^2 + \gamma^2}.$$

We can define a hash mapping by taking samples from a Cauchy distribution. Our hashing function is defined as follows.

Definition. Let A be an $M \times N$ matrix such that its components are independent and randomly generated according to a Cauchy(γ) distribution, and U be a vector with M random independent components distributed uniformly between 0 and 2. We define the transformation $H_{A,U} : \mathbb{R}^N \to \{-\frac{1}{\sqrt{M}}, \frac{1}{\sqrt{M}}\}^M$ as

$$H_{A,U}(\mathbf{x}) = \frac{1}{\sqrt{M}}\, h(A\mathbf{x} + U),$$

where h is taken component-wise and it is given by

$$h(t) = 2 \cdot (\lfloor t \rfloor \pmod 2) - 1\,.$$

The periodic function h of period 2 is illustrated in Figure 1 and can take values of either 1 or -1.

Thus, this transformation takes a real value vector with N components and gives a vector with M components, which can be either $\frac{1}{\sqrt{M}}$ or $-\frac{1}{\sqrt{M}}$. This definition is needed to ease the computation between hashes for approximating a kernel. However, as we will see in this paper, since we have a binary vector, each component can be encoded using just one bit reducing the overall size of the hashed data. The definition of our hash is similar to that provided in [14] where a Gaussian distribution is used instead of a Cauchy.

In order to see the utility of this hashing function, we can analyze the inner product between hashes. Since we are working with a random transformation, the next theorem provides an expression for its expected value.

Theorem 1. $\forall \mathbf{x}_1\,,\, \mathbf{x}_2 \in \mathbb{R}^N$, the expected value of inner product between their corresponding transformed points is:

$$\mathbb{E}\Big(\big\langle H_{A,U}(\mathbf{x}_1)\,,\, H_{A,U}(\mathbf{x}_2)\big\rangle\Big) = \frac{8}{\pi^2} \sum_{i=1}^{\infty} \frac{1}{(2i-1)^2} \exp\big(-\pi\,\gamma\,(2i-1)\|\mathbf{x}_1 - \mathbf{x}_2\|_1\big)\,.$$

This expression only depends on the ℓ_1 distance between the original vectors $\mathbf{x}_1$ and $\mathbf{x}_2$. Figure 2(a) shows the shape of these curves for fixed values of γ. It is easy to prove that this function is decreasing and tends to 0 when $\|\mathbf{x}_1 - \mathbf{x}_2\|_1$ tends to infinity. In addition, if $\mathbf{x}_1 = \mathbf{x}_2$, we can see that the previous expression is 1.

The expression given in Theorem 1 is not exactly the expression for the Laplacian Kernel, which is properly defined as follows:

$$K(\mathbf{x}_1\,,\,\mathbf{x}_2) = \exp\big(-\gamma\,\pi\,\|\mathbf{x}_1 - \mathbf{x}_2\|_1\big),$$

where γ is a parameter of the model.

However, we can find bounds which depend on this kernel. The following proposition shows an upper and lower bound for the expected value of the inner product between hashes.

Proposition 1. If A and U follow the hypothesis of Theorem 1, we have

$$\frac{8}{\pi^2} \exp\big(-\pi\,\gamma\|\mathbf{x}_1 - \mathbf{x}_2\|_1\big) \leq \mathbb{E}\Big(\big\langle H_{A,U}(\mathbf{x}_1)\,,\, H_{A,U}(\mathbf{x}_2)\big\rangle\Big) \leq \exp\big(-\pi\,\gamma\|\mathbf{x}_1 - \mathbf{x}_2\|_1\big) \quad (1)$$

These inequalities show how similar the Laplacian Kernel is compared to the expression given by Theorem 1. We can see that the difference between these two kernels is bounded by $\frac{\pi^2-8}{\pi^2} \exp(-\pi\gamma\|\mathbf{x}_1 - \mathbf{x}_2\|_1)$, which means that the difference decreases as $\|\mathbf{x}_1 - \mathbf{x}_2\|_1$ increases, converging to 0. Actually, in the worse case, this is when $\|\mathbf{x}_1 - \mathbf{x}_2\|_1 = 0$, the difference between these kernels is $\frac{\pi^2-8}{\pi^2} \approx 0.189$.

Nevertheless, both Theorem 1 and Proposition 1 are in expectation and eventually the actual result given by the stochastic hashing may differ. However, thanks to the Law of Large Numbers, the Inner Product between $H_{A,U}(\mathbf{x}_1)$ and $H_{A,U}(\mathbf{x}_2)$ converges to the expected value for large values of M. In fact, at the moment of computing the inner product between two hashes, we are performing an operation which is a sum of IID random variables, divided by M, to obtain a simple average of random variables. Figure 2(b) shows a simulation, where the empirical value is plotted as a function of the actual ℓ_1 distance between vectors. We see here that the empirical estimate closely follows the plot for the theoretical expression in figure 2(a).

A more formal guarantee is given by Theorem 2. based on Hoeffding's inequality, where the probability of having a small error estimation depends on the value of M and the number of points to transform involved.

Theorem 2. If we have n points $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathbb{R}^N$, and $M \geq \frac{4}{\varepsilon^2} \log\left(\frac{n}{2\eta}\right)$, then

$$\mathbb{P}\Big(\forall \mathbf{x}_i, \mathbf{x}_j \,, \Big| \langle H_{A,U}(\mathbf{x}_i)\,, H_{A,U}(\mathbf{x}_j)\rangle - \mathbb{E}\left(\langle H_{A,U}(\mathbf{x}_i)\,,\, H_{A,U}(\mathbf{x}_j)\rangle\right)\Big| < \varepsilon\Big) \;\geq\; 1-\eta.$$

This result gives a way to measure the quality on the approximation made by the hashes. In particular, if η and ε are fixed, the number of components needed in the Hashing function depends on the log of the number of points n that we want to transform. Moreover, note that the value of M does not depend on the dimensionality of the input data. Both facts may have important implications when we work with high dimensional data.

2.2. One-bit Representation and Inference

As we have seen, the function $H_{A,U}$ takes a real valued vector $\mathbf{x}$ and outputs a vector with M components which its values can be either $-\frac{1}{\sqrt{M}}$ or $\frac{1}{\sqrt{M}}$. This means, that even though M may be smaller than the original data dimension, the final representation of each hash component is a float number.

However, we can redefine the hashing scheme and the operation between the hashes in order to get the same approximation we have already established. In fact, given a $M \times N$ matrix A and the M-dimensional vector U, we can define

$$Q_{A,U}(\mathbf{x}) = \lfloor A\mathbf{x} + U \rfloor \pmod 2.$$

In this case, the function $Q_{A,U}$ outputs a vector with M components, which its values can be either 0 or 1. Hence, each component can be represented with just one bit. Therefore, after applying this transformation, each vector will be mapped to a string with M bits.

In addition, it is quite straightforward to prove that the inner product $H_{A,U}(\mathbf{x}_1)$ and $H_{A,U}(\mathbf{x}_2)$ can be recovered using $Q_{A,U}(\mathbf{x}_1)$ and $Q_{A,U}(\mathbf{x}_2)$. Indeed, we have the next equation

$$\langle H_{A,U}(\mathbf{x}_1)\,,\, H_{A,U}(\mathbf{x}_2)\rangle \;=\; 1 - \frac{2}{M}\sum_{i=1}^{M} Q_{A,U}(\mathbf{x}_1)_i \oplus Q_{A,U}(\mathbf{x}_2)_i\,,$$

where $\oplus$ is the XOR operation between bits. Thus, we can recover the kernel approximation using the bit representation and the Hamming distance between hashes. The natural cost is the cost of computing the approximation using an inner product operation. Nevertheless, the Hamming distance computation is not expensive compared to other kind of operations.

2.3. Hashing Trick - Hashing Applied to SVM

Commonly, SVM models are trained using its dual form that naturally leads to the kernel trick. In fact, its dual form only depends on the inner product between pairs of input vectors. Thus, in order to find non-linear decision boundaries, we try to find a linear boundary in a higher dimensional space mapping the input data using some function ϕ. Hence, instead of working directly with $\langle \phi(\mathbf{x}), \phi(\mathbf{y})\rangle$, the kernel trick defines a function K such that $K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y})\rangle$. Thus, we can work in a higher dimensional space without an explicit mapping.

The kernel trick implies the computation of the Kernel matrix using the input features and a non-linear kernel. However, such computation scales poorly because its time and storage complexity are quadratic in the dimensionality and number of the input features. Hence, we can apply our hashing scheme to the SVM. First, we take the input features and generate our hashes as described in the previous sections. Then, we use the hashes to compute the kernel matrix with the inner product, which is a linear operation. This combination approximates a non-linear Laplacian kernel SVM. Formally

$$K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y})\rangle \approx \langle H(\mathbf{x}), H(\mathbf{y})\rangle\,.$$

We can name this procedure *Hashing Trick* [22].

3. EXPERIMENTAL SETUP AND RESULTS

The section aims to show the benefits of using hashing plus a linear SVM over the input features plus the non-linear SVM with Laplacian kernel. The experiments are in the context of the DCASE Task 1 - Acoustic Scene Classification [21]. The pipelines of the experiments are also illustrated in Figure 3.

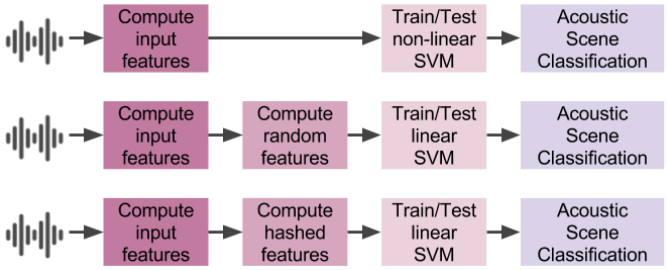


Fig. 3. The audio recordings are used to extract input features. Then, these features are used to train an SVM in three different ways. One is to pass the features directly to a non-linear SVM, second and third are to compute random features [13] or hashes and pass them to a linear SVM. Lastly, the trained SVM is used for classification.

3.1. Acoustic Scenes Data Set

For our experiments we used the "DCASE: TUT Acoustic Scenes 2017" development dataset [23]. It consists of recordings from various acoustic scenes of 3-5 minutes long. The 15 acoustic scenes are: *Bus, Cafe / Restaurant, Car, City center, Forest path, Grocery store, Home, Lakeside beach, Library, Metro station, Office, Residential area, Train, Tram, Urban park.*

3.2. Audio Feature Extraction

We extracted the large set of audio features of 6,553 dimensionality employed in [3, 6]. The set includes different features to extract different relevant information from the acoustic scenes, which consist of multiple sound sources. The set is extracted using the toolkit openSMILE [24] with the configuration file *emolarge.conf*. The features are divided in four categories: cepstral, spectral, energy related and voicing features, and are extracted every 10 ms from 25 ms frames. Moreover, we included functionals, such as mean, standard deviation, percentiles and quartiles, linear regression functionals, or local minima/maxima.

3.3. Acoustic Scene Classification

First, we evaluate the input features using the non-linear SVM with the Laplacian kernel and then with the Series kernel. Then, we evaluate the hashes with a linear SVM to approximate the baselines.

Method \ γ	2^{-12}	2^{-14}	2^{-16}	2^{-18}	2^{-20}
Laplacian Kernel	71.21	77.91	**78.64**	78.51	78.51
Series Kernel	70.30	77.10	78.30	78.49	**78.55**
Hash ($M = 2^6$)	19.80	40.59	37.75	23.71	13.00
Hash ($M = 2^8$)	31.28	53.95	50.96	44.00	28.94
Hash ($M = 2^{10}$)	49.85	68.44	66.28	58.11	46.95
Hash ($M = 2^{12}$)	64.45	74.86	75.18	70.51	64.92
Hash ($M = 2^{14}$)	68.57	76.06	76.31	75.93	72.90
Hash ($M = 2^{16}$)	69.70	76.72	**77.55**	77.38	76.21

Table 1. Accuracy performance (%) as γ varies for the Laplacian kernel (with input features), Series kernel (with input features) and different hashing dimensionality M in bits (with linear SVM). The best results for each type are in bold. Note how as the value of M increases the performance is better approximated, but the dimensionality increases.

Lastly, we include from our previous work [13] the random features with a linear SVM to approximate the Laplacian kernel.

We trained SVM classifiers with a one-vs-all setup and tuned the γ parameter. For each experiment we used the feature vectors of the testing set to evaluate performance and measured it with accuracy.

3.3.1. Baseline

The baseline performance consisted of two experiments, the input features using the SVM with Laplacian kernel and with Series kernel. Even though we have shown how similar both kernels are, we wanted to see how this holds in our ASC task. Hence, we used the acoustic feature vectors of 6,553 components and trained SVMs for each type of kernel.

3.3.2. Hashing

In the second set of experiments, we used our hashing scheme and tried different hash sizes (M) to compare against the baseline performance. Hence, we used the training and testing acoustic feature vectors of 6,553 dims to compute hashes. The size of the hashes was varied from $M = 2^6$ to $M = 2^{16}$ bits. The hashes were used to train the SVM with a linear kernel.

3.4. Results on Acoustic Scenes Classification

3.4.1. Baseline

The baseline accuracy results for both kernel types, Laplacian and Series, performed similarly as we expected and depending on the value of the parameter γ the variation between both kernels was closed. Also, note that the variation of γ affected the range of performance of both kernels. First, the Laplacian Kernel had a range of 71.21% ($\gamma = 2^{-12}$) to 78.64% ($\gamma = 2^{-16}$). Second, the Series Kernel had a range of 70.30% ($\gamma = 2^{-12}$) to 78.55% ($\gamma = 2^{-20}$). Details of these results are presented in Table 1.

3.4.2. Hashing

The accuracy results of the hashes improved as the size M increased to a number close to the baselines. Similar to the baseline experiments, the performance varied depending on the value of the parameter γ. For the best performing hash and largest size (2^{16}) the range was 69.70% ($\gamma = 2^{-12}$) to 77.55% ($\gamma = 2^{-16}$), as Table 1 shows.

4. DISCUSSION

We discuss the different methods considered in this paper and in the related work. Particularly, in Table 2 we include the official DCASE

Method	Accuracy	# of Bits
DCASE Challenge [21]	74.8%	-
Series Kernel	78.5%	$> 2^{18}$
Laplacian Kernel	78.6%	$> 2^{18}$
Random features $M = 2^{12}$ [13]	75.8%	2^{18}
Hashing $M = 2^{16}$	77.5%	2^{16}
Hashing $M = 2^{12}$	75.2%	2^{12}

Table 2. The performance of the hashing scheme outperforms the reported Challenge score and the random features. Moreover, performance is comparable to using the Laplacian Kernel, however with the benefits of reducing the bit representation by 4 times.

performance based on a Multilayer Perceptron, our baselines, our hashing scheme and the random features [13].

First, we can see that the difference between the Laplacian and the Series Kernel is minimal. Even though both have different expressions, in practice the performance is similar and hence, supporting the validity of the hashing approximation. In addition, we can see that after applying the hashing technique we have a minimal loss of performance with the advantage of working with a linear problem instead of a non-linear quadratic one.

The most significant findings are related to the size of the output obtained from our hashing function. In this experiment, each input sample is represented by a vector of 6,553 components, which is more than 2^{18} bits if each component is represented by 8 bytes. In contrast, to achieve similar results with our hashing technique, we need 2^{16} bits, that is, 4 times reduction in size. Hence, this method not only change a nonlinear problem to a liner one, but also reduces the dimensionality of the feature vector.

A similar technique, random features [13], employs 2^{12} components, which allowed us to approximate the Laplacian kernel with an accuracy of 75.8%. Even though the dimensionality of the random features was 3 times lower than the original vector, each random feature used 2^{18} bits if each component is represented by 8 bytes. In contrast, using the hashing technique, with 2^{14} bits we obtain a performance of 76.31%. Hence, we can reduce the size of the final representation by 16 times and obtain comparable results. Actually, with a small loss of performance, we may use hashes with 2^{12} bits and obtain an accuracy of 75.18%, reducing the size of the final representation by 64 times or 6 powers of two.

Our hashing has several potential applications for acoustic scenes and sound events. For instance, reduce the struggle of mobile devices with larger datasets and larger size of embeddings from neural networks. Another example is related to cloud computing and privacy. If we need to process sensitive information in the cloud, we can apply our hashing technique keeping the parameters A and U private. Thus, we can still process ans classify the hashed recordings in the cloud because the hashing hides the information without revealing the actual content.

5. CONCLUSIONS

In this paper we have presented a hashing scheme that allows us to approximate the Laplacian kernel between features through the inner product between hashes. Our main contribution was to show the theoretical guarantees to validate this approximation. We studied this scheme with SVMs for Acoustic Scene Classification. The hashing provides minimal degradation of performance, as well as employs binary representation, reducing up to six powers of two the storage. This has significant implications in the big data context, where high dimensional features and large datasets must be processed.

6. REFERENCES

[1] D. Giannoulis, E. Benetos, D. Stowell, M. Rossignol, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: an IEEE AASP challenge," in *2013 IEEE WASPAA*. IEEE, 2013, pp. 1–4.

[2] Z. Zhang and B. Schuller, "Semi-supervised learning helps in sound event classification," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 333–336.

[3] J. T. Geiger, B. Schuller, and G. Rigoll, "Large-scale audio feature extraction and SVM for acoustic scene classification," in *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2013, pp. 1–4.

[4] F. Metze, S. Rawat, and Y. Wang, "Improved audio features for large-scale multimedia event detection," in *Multimedia and Expo (ICME), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1–6.

[5] B. Elizalde, A. Kumar, A. Shah, R. Badlani, E. Vincent, B. Raj, and I. Lane, "Experiments on the DCASE Challenge 2016: Acoustic scene classification and sound event detection in real life recording," in *DCASE2016 Workshop on Detection and Classification of Acoustic Scenes and Events*, 2016.

[6] Z. Zhang, D. Liu, J. Han, and B. Schuller, "Learning audio sequence representations for acoustic event classification," *arXiv preprint arXiv:1707.08729*, 2017.

[7] R. Arandjelović and A. Zisserman, "Look, listen and learn," *arXiv preprint arXiv:1705.08168*, 2017.

[8] Y. Aytar, C. Vondrick, and A. Torralba, "Soundnet: Learning sound representations from unlabeled video," in *Advances in Neural Information Processing Systems*, 2016, pp. 892–900.

[9] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, "Recent advances of large-scale linear classification," *Proceedings of the IEEE*, vol. 100, no. 9, pp. 2584–2603, 2012.

[10] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in neural information processing systems*, 2008, pp. 1177–1184.

[11] F. Li, C. Ionescu, and C. Sminchisescu, "Random fourier approximations for skewed multiplicative histogram kernels," in *DAGM 2010: Pattern Recognition pp 262-271*, 2010.

[12] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," in *IEEE Transactions on Pattern Analysis and Machine Intelligence, 3(34):480492, 2012*, 2012.

[13] A. Jimenez, B. Elizalde, and B. Raj, "DCASE 2017 task 1: Acoustic scene classification using shift-invariant kernels and random features," DCASE2017 Challenge, Tech. Rep., 2017.

[14] P. T. Boufounos and H. Mansour, "Universal embeddings for kernel machine classification," *SampTA*, 2015.

[15] A. Sinha and J. C. Duchi, "Learning kernels with random features," in *NIPS*, 2016.

[16] Z. Lu, A. May, K. Liu, A. B. Garakani, D. Guo, A. Bellet, L. Fan, M. Collins, B. Kingsbury, M. Picheny, *et al.*, "How to scale up kernel methods to be as good as deep neural nets," *arXiv preprint arXiv:1411.4000*, 2014.

[17] T. Mikami, Y. Kojima, K. Yonezawa, M. Yamamoto, and M. Furukawa, "An SVM-based classification of oral and nasal snoring sounds with kullback-leibler kernel," in *SICE Annual Conference (SICE), 2012 Proceedings of*. IEEE, 2012, pp. 1795–1797.

[18] ——, "Spectral classification of oral and nasal snoring sounds using a support vector machine," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, pp. 611–621, 2013.

[19] J. Yang, V. Sindhwani, Q. Fan, H. Avron, and M. Mahoney, "Random Laplace feature maps for semigroup kernels on histograms," *CVPR*, 2014.

[20] K.-S. Goh, E. Chang, and K.-T. Cheng, "Support vector machine pairwise classifiers with error reduction for image classification," in *Proceedings of the 2001 ACM workshops on Multimedia: multimedia information retrieval*. ACM, 2001, pp. 32–37.

[21] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, submitted.

[22] J. L. A. S. Kilian Weinberger, Anirban Dasgupta and J. Attenberg, "Feature hashing for large scale multitask learning," in *ICML*, 2009.

[23] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," in *24th European Signal Processing Conference 2016 (EUSIPCO 2016)*, Budapest, Hungary, 2016.

[24] F. Eyben, M. Wöllmer, and B. Schuller, "Opensmile: the munich versatile and fast open-source audio feature extractor," in *Proceedings of the 18th ACM international conference on Multimedia*. ACM, 2010, pp. 1459–1462.