Acta Univ. Sapientiae Informatica **14,** 1 (2022) 1–21

DOI: 10.2478/ausi-2022-0001

Using the Bag-of-Audio-Words approach for emotion recognition

Mercedes VETRÁB
University of Szeged, Institute of Informatics
Hungary, Szeged, Árpád tér 2.
ELKH-SZTE Research Group on Artificial Intelligence
Hungary, Szeged, Tisza Lajos körút 103.
orcid=0000-0001-7511-2910
email: vetrabm@inf.u-szeged.hu

Gábor GOSZTOLYA
University of Szeged, Institute of Informatics
Hungary, Szeged, Árpád tér 2.
ELKH-SZTE Research Group on Artificial Intelligence
Hungary, Szeged, Tisza Lajos körút 103.
orcid=0000-0002-2864-6466
email: ggabor@inf.u-szeged.hu

Abstract. The problem of varying length recordings is a well-known issue in paralinguistics. We investigated how to resolve this problem using the bag-of-audio-words feature extraction approach. The steps of this technique involve preprocessing, clustering, quantization and normalization. The bag-of-audio-words technique is competitive in the area of speech emotion recognition, but the method has several parameters that need to be precisely tuned for good efficiency. The main aim of our study was to analyse the effectiveness of bag-of-audio-words method and try to find the best parameter values for emotion recognition. We optimized the parameters one-by-one, but built on the results of each other. We performed the feature extraction, using openSMILE. Next we transformed our features into same-sized vectors with openXBOW, and finally trained and evaluated SVM models with 10-fold-crossvalidation and UAR. In our experiments, we worked with a Hungarian emotion database. According to our results, the emotion classification performance improves with the bag-of-audio-words feature representation. Not all the BoAW parameters have the optimal settings but later we can make clear recommendations on how to set bag-of-audio-words parameters for emotion detection tasks.

Computing Classification System 1998: H.3.1, I.2.7.
Mathematics Subject Classification 2010: 68R15
Key words and phrases: bag-of-audio-words, emotion detection, human voice, sound processing

1 Introduction

Human speech is not only used for encoding the words uttered, but it also contains other information about the speakers. For example, about their physical and mental state. These include the emotional state, signs of illness, depression, joy and so on. This extra information can be used in various ways by computer science and engineering information technology. Nowadays emotion detection from audio data (speech emotion recognition or SER) is an active area of research with a wide range of possible applications. It can be used in the human-computer interfaces, like that for monitoring human communication [12] and detecting the gender of the speaker, or their emotional state, or how confident they are. We can also use paralinguistics in dialogue systems [3] where we can detect the problematic dialogue phrases or adapt the dialogue to help the speaker. Besides this, it may be useful in healthcare systems [10, 26] to monitor the patient's mental state. Last, but not least we can utilize emotion detection in call centres [26]. For instance if the client get angry, we can automatically inform a boss about this. Using machines for emotion recognition and monitoring systems is a currently evolving area. In the future with good emotion recognition systems, we will be able to create more human-oriented and friendlier systems. For example we can create intelligent tutorial systems that can adapt to the student's mental state and give them more constructive advice. In addition, we can use it for lie detection to improve law enforcement. Emotion detection is also useful in a call centre or a banking software monitoring application, where we can monitor how patient members of the staff are. Furthermore we can also use it for the support diagnostics of therapists, create more empathic healthcare robots, and in computer games use it to set the difficulty of the game by the emotion of the user [13]. There are other interesting applications in paralinguistics. Human computer interfaces and user adaptation systems could be used to recognizing the age and the gender of the speaker from their voice. For instance here are some electronic systems that can use these human features: an automatic dialogue system can adapt to the speaker by speaking slower and louder for an older user or use a different corpus for younger and older customers; an interactive voice response system can choose the background music by guessing the age and the gender of the user; smart home systems can adapt to the age of the speaker because an older customer needs more automation while a younger customer need a more collaborative system; a police call analysis system can identify the age and the gender of a suspect from a telephone call [17].

Since the beginning of research in this area, many feature extraction and classification techniques have been used along with different datasets to get the best results. The variable length of recordings has always been a big problem here. This is due to the fact that our recordings have different lengths, but our classifiers expect a fixed length input. So one of the most difficult problems in speech emotion recognition and in other paralinguistics areas is feature extraction, because as we mentioned our recordings are different in length, but the classification techniques requires fixed-sized feature vectors. Several methods have already been developed to tackle the problem of varying length and to make the features extracted from the recordings the same length. For example x-vectors [18], i-vectors [27], Fisher vectors [8], neural networks [9] and the Bag-of-Audio-Words (i.e: BoAW [15]) approach that we investigate here. Our experiments were performed on a Hungarian database and our final results indicated that the BoAW technique can be used effectively. However, we should also add that creating any BoAW feature representation is sensitive to the parameter settings and working with bigger codebooks for better classification result requires more CPU time.

Our baseline comes from a Hungarian emotion speech database. Previous studies (i.e.: [25], [23], [8]) using this database produced accuracy scores of 66–70%. Previous results in speech emotion recognition with another databases are came between 60–80% [24, 11, 20]. Our results give us an unweighted average recall (UAR) score of 66–71%.

2 Bag-of-audio-words method

Using the bag-of-audio-words feature representation, we can overcome the above-mentioned problem of varying length. This feature extraction method is similar to the bag-of-visual-words [5] and the bag-of-words [28] techniques, which are used in image and speech preprocessing. Now we will present the BoAW workflow.

BoAW first performs an analysis on the entire audio database and then, based on the results obtained, generates statistics for each file separately that represent their relationship to the entire database. Figure 1 shows the general workflow for generating a BoAW representation from a dataset. The two columns belong to the training and the test set extraction steps. As we can see, the extraction of the test set depends on the train set extraction workflow, but it mostly contains the same steps, so let us discuss the training set.

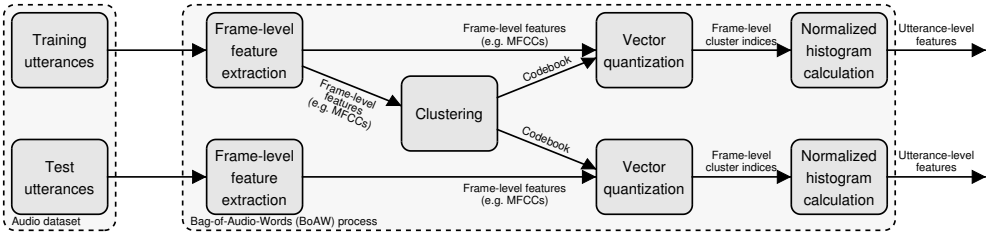

Figure 1: Workflow for the bag-of-audio-words technique.

In the first step, we have to extract the frame-level feature vectors per recording. In this step we get a different number of feature vectors for each recording, because the number of vectors depends on the original length of the evaluated recording and the frame's windowing size. In the next step, we work with all the feature vectors from all the recordings, collect them into one big "bag" and perform clustering on it. The purpose of this is to break down the vectors of the "bag" into meaningful subsets such that vectors in the same groups are more similar to other vectors in the same group than those in other groups. The number of clusters to be produced is determined by us. This cluster size parameter N is one of the parameters of the BoAW method. The centres of the created clusters will be called "codewords". The group of these "codewords" will be the "codebook". The N parameter is called the codebook size. The vector dimension of the classification will depend on the codebook size.

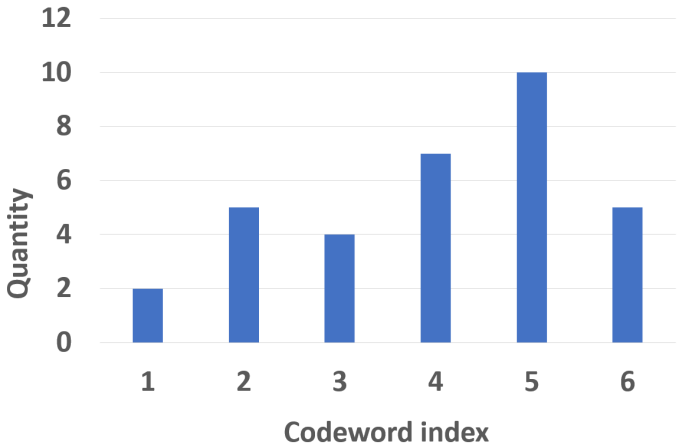

Figure 2: A bag-of-audio-words histogram of the recording.

After the clustering process, in the vector quantization step, we again work with individual recordings and create a histogram for each recording. We replace the original feature vectors by the index of the closest codeword. We then find the nearest neighbours using the minimum Euclidean distance. We can also specify how many closest vectors we are looking for (this is also a parameter of the BoAW method). So this produced, same-sized (i.e. N) histograms for each recording. The x-axis of the histogram lists the index of each codeword, and the y-axis shows quantities which represent how many of the feature vectors of the recording were mapped into a particular codeword. After quantization, each file's feature becomes independent of the length of the particular audio recording. For example, Figure 2 shows a histogram for one recording. In this case the codewords are represented by their indices (i.e. $1, 2, 3$ and so on). This recording has 43 frames, and every frame gets mapped into 1 codeword.

In the last step, we normalize the histogram, so the given frequencies are divided by the number of frames of the speech recording. We notice that each histogram can be represented by a codebook-sized vector. These histograms will be our new feature vectors that have an independent length from the recording sizes. We will call this set of histograms "Bag-of-Audio-Words" and use it as features for our classifier.

Figure 1 shows how the clustering step can be omitted for the test set. This can be done because the openXBOW software allows us to save two important things: the parameter settings applied to the training set and the computed codebook. Then we use these later in the test set, so that the quantization step can be performed on the cluster centres generated during the training set recordings without re-clustering. This operation is easy to implement, since the test file has frame-level feature vectors. They are the same as those produced with a set of features like the vectors of the training set, so we can classify them into each cluster based on their distance from the previously defined codewords.

2.1 Parameters of the BoAW method

The BoAW method has many adjustable parameters that can influence the process of codebook creation. In our study, we tested the effect of the preprocessing method, clustering method, the codebook size N, and the quantization neighbour number parameters on the learning algorithm performance. For the codebook building, we used an open-source program called openXBOW [21].

Preprocessing techniques: with openXBOW we can do some preprocessing for the frame level descriptors, before the clustering step. If some of the features have extremely high or low values compared to the others, they may dominate the Euclidean distance during the BoAW vector quantization step. We tested to see how preprocessing improves the performance, so we tried out three different solutions for it. The first one was without any preprocessing, the second one was normalizing the feature vectors and the third one was by standardizing the vectors before clustering.

Clustering method: One important factor is the clustering procedure used to create the codebook. Pancoast and Akbacak used k-means in their original study [16]; however, due to the large number of frames to be clustered, the runtime of this approach is very high. Rawat et al offer simple random sampling [19]; its runtime is marginally better than the k-means, and it does not really affect the performance. Later, Arthur et al. applied k-means++ clustering [1], a cluster center initialization procedure, which was used instead of completely random sampling, hence the distribution of cluster centers became more balanced. Compared to k-means, cluster centers are not selected at random during initialization, but selected via a uniform distribution. We tested the effect of applying the k-means and k-means++ methods on our data.

Histogram neighbour number: Instead of looking for just the closest codeword, each vector may also be assigned to a certain number of the closest codewords. Pancoast and Akbacak assume that instead of just using the closest cluster to each frame, we can assign a set of closest neighbours [16]. This leads to a more precise description of the recordings with the same feature vector size. This is why we experimented with two different settings (5 and 10).

Codebook size: As we said earlier, we can control how many clusters we wish to create, and how long we want the feature vectors to be. In each experiment we tested the effect of the following lengths: 32, 64, 128, 256, 512, 1 024, 2 048, 4 096, 8 192.

Derivatives: In speech processing, it is common practice to subtract the first and second derivatives of the feature vectors extracted from the sound recordings. These are the so-called deltas and delta-deltas, from which the dynamics of speech can be deduced [6]. With the help of the openXBOW program, we can create separate codebooks for the original low-level descriptors and another for the Δs.

3 Data and methods

Next, we will present our experimental setup and environment: the database, the classification method and its parameters, the evaluation metric, and the feature set we used.

3.1 Hungarian emotion database

In each experiment, we created and evaluated our classification model on the Hungarian emotion database. It contains utterances of 97 native Hungarian and Hungarian-speaking speakers [25]. The voice samples were recorded during television shows. The vast majority of segments were recorded from an emotion-rich, continuous, spontaneous programme with actors. The other part came from an improvisation entertainment show. In the first case due to the acting performance, the samples are vivid, and the emotions are clearer. The samples from the second case due to the improvisation, are closer to real-life emotions. The database contains 1111 sentences, which were separated into an 831 sample training set and a 280 sample test set. We had to detect four emotions, namely neutral, joy, anger, sad. The distribution of the emotions however was not uniform. The training set sample distribution was: $\approx$ 57% neutral, $\approx$ 6% sad, $\approx$ 9% joy and $\approx$ 27% anger. The test set sample distribution was: $\approx$ 62% neutral, $\approx$ 4% sad $\approx$ 7% joy and $\approx$ 27% anger. The training set contains approximately 20 minutes of recordings and the test set contains approximately 7 minutes of recordings. The sampling frequency of the samples is 16 kHz. Earlier studies working with the same database were able to achieve a classification accuracy score of 66–70% [23, 25, 8].

3.2 Feature set

The feature set employed in the study came from the INTERSPEECH 2013 Paralinguistic Challenge [22]. It contains 65 frame-level features: 55 spectral; 6 voicing related low-level descriptors; 4 energy-related. 60 ms frame (Gaussian window function) and a sigma value of 0.4 was used for the speech-related features; and a 25 ms frame (Hamming window function with a step size of 10 ms) for the others.

For feature extraction we used the open-source openSMILE software package [7] with the `IS13 ComParE` config file. The final feature set we used contained not only the basic features, but also their derivatives. We used deltas because we wanted to get information about the dynamics of the speech samples over time.

3.3 Evaluation method

The classification was performed using the LIBSVM library [4], which is an SVM (Support Vector Machine [2]) implementation written in C++; here we used the Python extension. The SVM C complexity parameter was tested in the range 10^{-5} to 10^{0}. In the evaluated configurations the following powers of 10 used were: −5; −4; −3; −2; −1 and 0. We applied a Python implemented standardization on the input BoAW feature representations before each model was trained.

In the optimization part of our experiments, we worked with the training set, based on 10-fold cross-validation. We split the data into roughly 10 equal folds, where each speaker is shown in only one fold, so each fold became absolute speaker independent. Afterwards, we trained on the 9/10 part and evaluated on the 1/10 part for each possible combination. Consequently, when evaluating one part as a test set, we got predictions for a specific part of the entire database which did not overlap with the other parts, so after running all possible combinations, we had exactly one prediction for each element of the entire database. UAR metrics could then be easily derived from this. After the 10th evaluation, we collected the predicted percentage scores from all the test cases (one score for each sample) and calculated the UAR metrics. The unweighted average recall was used as an indicator to see how good the actual feature set was for emotion recognition.

In the test scenario, we trained a model on the whole training set with the optimal C parameter value found above and evaluated it on the test set with the Unweighted Average Recall (UAR) metrics [14]. The reason we use this metric, because we have imbalanced classes. Accuracy and UAR metrics are related, but the accuracy gives a more optimistic value because it gives higher scores to classes with more samples, but the UAR gives the correct expectation on each class predictions.

In the last part, we describe our experimental procedure and the evaluation of our results. We extracted 2×65 features (65 frame-level features and their derivatives) in a frame-level window. Therefore we created two codebooks in parallel (one for 65 frame-level features and one for their derivatives). Because of this, the codebook sizes given in this section have to be multiplied by 2 to get the number of features currently used. The results of each test cases are shown below and the best results are given in tabular form for better transparency. In each figure, the x-axis shows the size of the codebook (which has to be multiplied), and the y-axis shows the UAR of the SVM. The legends of our figures contains abbreviations: “a1” means 1 neighbour during quantization; “a5”

Feature-preprocessing	**Maximum UAR**	**Codebook size**
No preprocessing	36.32%	8 192
Normalization	46.73%	4 096
Standardization	45.42%	1 024

Table 1: Preprocessing: The best results got without preprocessing, with normalization and standardization, when we evaluated our technique with cross-validation.

means 5 neighbours during quantization; "a10" means 10 neighbours during quantization; "standardized" and "stand" both means standardization during preprocessing; "normalized" and "norm" both means normalization during preprocessing; "k-means" means k-means clustering technique, "k-means++" means k-means++ clustering technique.

4 Tests and results

4.1 Preprocessing

In the first case, we compared preprocessing techniques before clustering. Preprocessing is always a good choice because databases contains outliers, which have a detrimental effect on learning effectiveness.

From our results (see Figure 3 and Table 1), it is apparent that the data without preprocessing proved to be the weakest in all cases. By comparison, normalization and standardization gave performance improvements that were nearly the same. Another advantage of normalizing or standardizing the input is that significantly fewer clusters are required for optimal performance than leaving the input unchanged (8 192). When we applied normalization, we got 46.35% for 1 024 codewords, so we found that in both normalization and standardization, a size of 1 024 was big enough to achieve the best performance. This lower codebook size also helps the performance of the SVM, because in a smaller feature space the speed and success of the learning will also increase.

The best result of the cross-validation (i.e. 46.73%) was achieved with normalization and a codebook size of 4 096. Otherwise there is no significant difference between the best standardization and normalization results. In addition, it is not clear that normalization or standardization will produce a better result with the feature set. Here, further tests were performed in parallel, with normalization and standardization to ascertain the benefits.

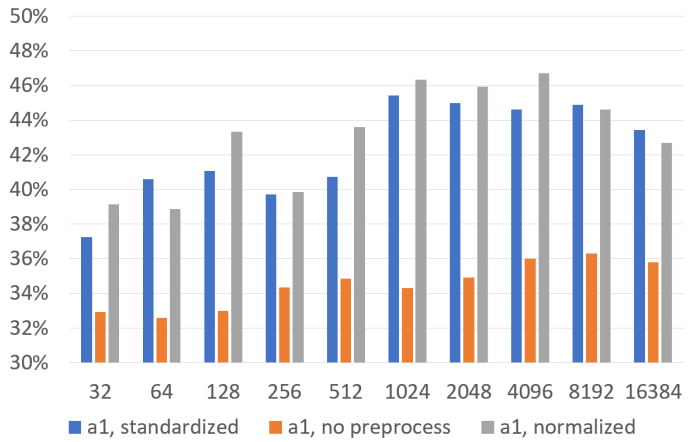

Figure 3: Preprocessing: The results obtained for different codebook sizes and preprocessing techniques.

Feature-preprocessing	**a**	**Maximum UAR**	**Codebook size**
Normalization	1	46.73%	4 096
	5	48.93%	4 096
	10	49.14%	16 384
Standardization	1	45.42%	1 024
	5	46.16%	8 192
	10	47.37%	8 192

Table 2: Number of neighbours: The best results obtained for $1, 5, 10$ neighbours with normalization and standardization.

4.2 Number of neighbours assigned during quantization

In the next comparison, we investigate how many closest codewords have to be assigned to a frame-level feature vector when creating a histogram, to achieve the optimal performance. In our experiments, we tested three options, where we used the closest $1/5/10$ neighbours. Based on the results of our previous optimization, all three quantatization options were also evaluated with normalization and standardization.

From our new results (see Table 2, Figure 5 and Figure 4), we may conclude that more than one neighbour gives better results in the majority of cases. This can be seen for both preprocessing techniques (normalization and standardization). As regards the performance of the classification algorithm, we

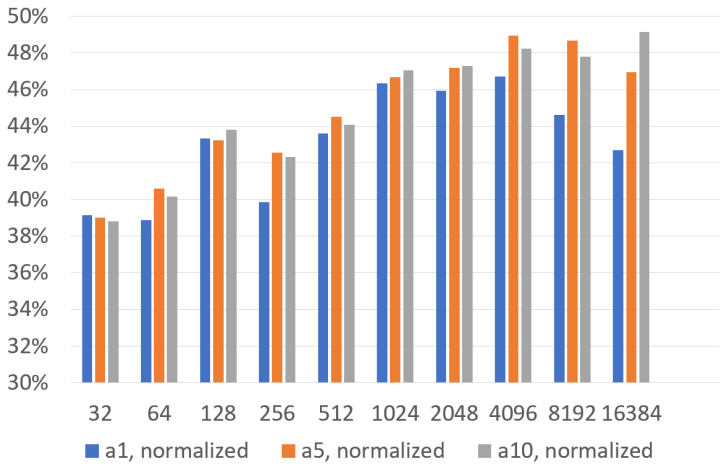

Figure 4: Number of neighbours: The results obtained for $1, 5, 10$ neighbours and normalization.

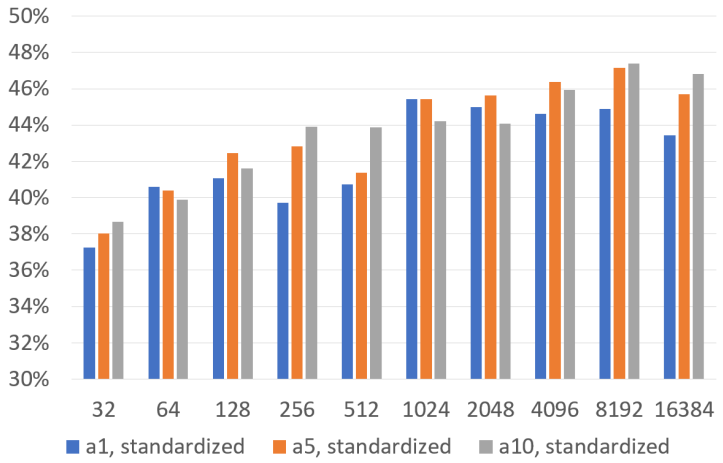

Figure 5: Number of neighbours: The results obtained for $1, 5, 10$ neighbours and standardization.

did not find any significant difference between the $a = 5$ and $a = 10$ values. As can be seen, above the codebook size of 512 we got significantly better results with applied preprocessing, so any kind of preprocessing is always a good choice if we want better results. With larger codebook sizes there is only a small difference (1%–3%) between the results got using standardization and normalization. Hence we need to investigate them further.

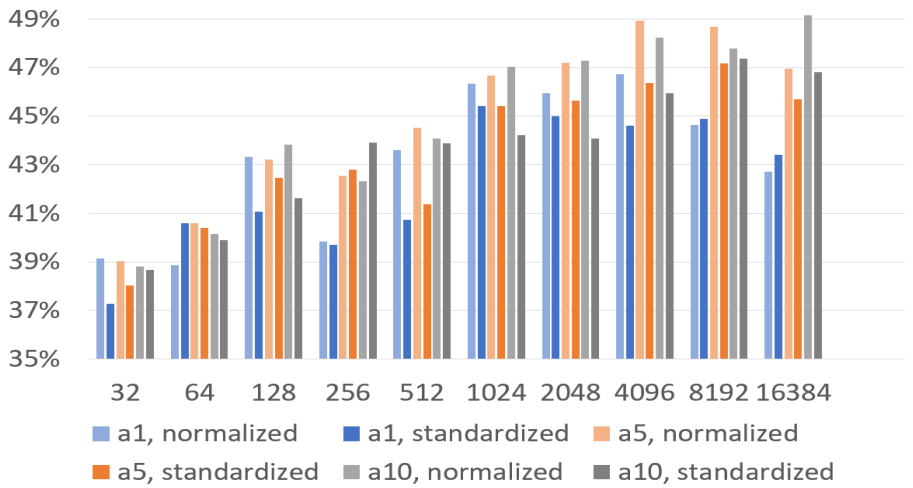

Figure 6: Number of neighbours: The results got using different codebook sizes, preprocessing techniques and neighbour counts.

Table 2 shows the best results for the different cases. We notice that 5 and 10 closest codewords give the same improvement, compared to the 1 neighbour version. Although not significant, the $a = 10$ option gives slightly better results in both preprocessing cases. Here we think that the multi-neighbour technique needs more clusters to achieve the best results. However it can be seen in Figure 6 that with standardization and normalization, the codebook size of 1 024 is already capable of giving results as good as the best single-neighbour variation. Based on our results, we decided in later test cases to test the 5 and 10 options in order to draw a more precise conclusion.

4.3 Clustering algorithm

The third parameter we investigated was the clustering algorithm, where we tested two techniques: k-means and k-means++. Based on our earlier results we decided to test them with normalization and standardization, and with 5 and 10 neighbours in the quantification step.
So our test cases were:

- 5 neighbours and standardization
- 5 neighbours and normalization
- 10 neighbours and standardization
- 10 neighbours and normalization

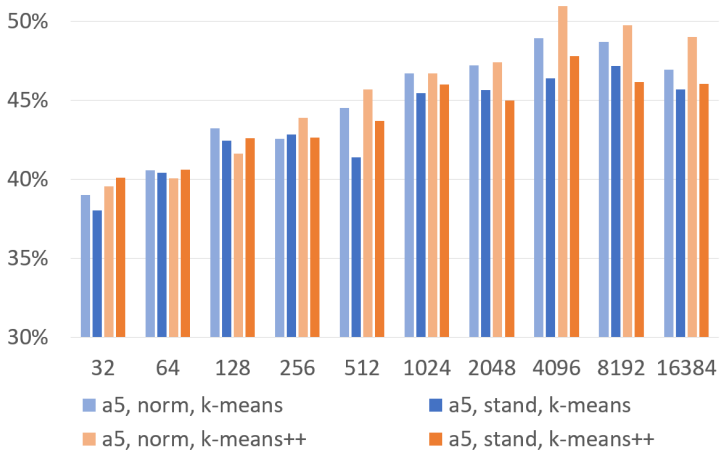

Figure 7: Clustering algorithm: The results obtained for k-means and k-means++ algorithms with 5 neighbours.

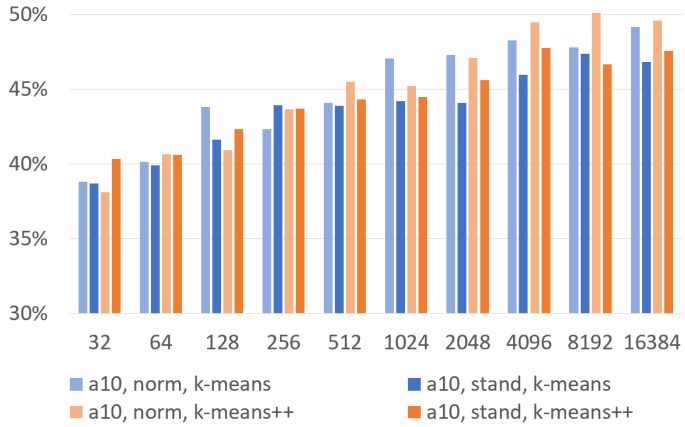

Figure 8: Clustering algorithm: The results obtained for k-means and k-means++ algorithms with 10 neighbours.

Based on the results (see Table 3, Figure 7, Figure 8, and Figure 9), we can say that both clustering methods have the same trend. Once again we get higher scores than 46% above a codebook size of 512. Also, we notice that normalization begins to perform better than the standardized case as the codebook size increases. The accuracy values are best with a codebook size of 4 096, which means that we need higher spatial dimensions to get better results. Since we did not find any significant difference between the trends of

Clustering-algorithm	Feature preprocessing	a	Maximum UAR	Codebook size
k-means	Normalization	5	48.93%	4 096
		10	49.14%	16 384
k-means	Standardization	5	46.16%	8 192
		10	47.37%	8 192
k-means++	Normalization	5	50.94%	4 096
		10	47.77%	4 096
k-means++	Standardization	5	50.08%	8 192
		10	47.74%	4 096

Table 3: Clustering algorithm: The best results for k-means and k-means++ algorithms with cross-validation.

k-means and k-means++, we decided to take the codebook sizes and settings that proved promising in our previous experiment (1 024 codebook size and normalization). Then, other tests were performed using the k-means algorithm.

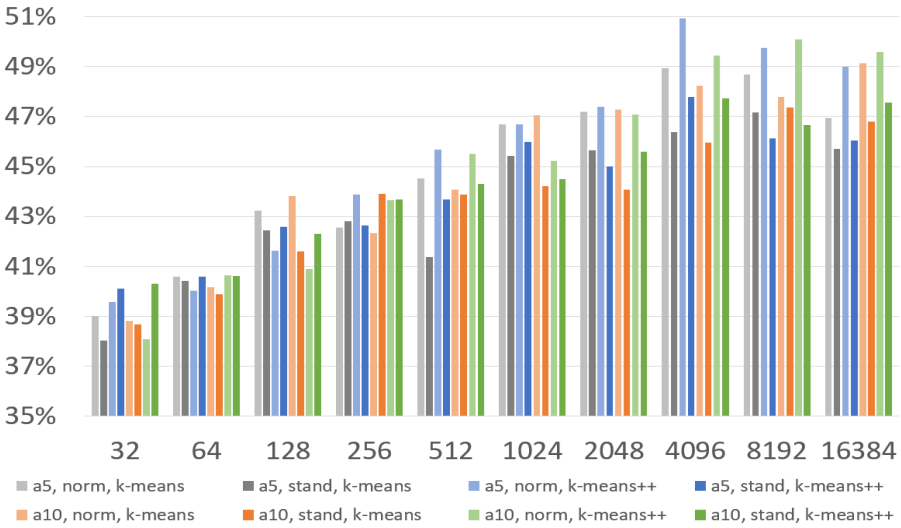

Figure 9: Clustering algorithm: The results got using different codebook sizes, preprocessing techniques, neighbour counts and quantatization algorithms.

4.4 Upsampling

Upsampling for smaller datasets and downsampling for large ones are common techniques when we have a very unbalanced dataset for labels. Because

Feature-preprocessing	**a**	**Maximum UAR**	**Codebook size**
Normalization	5	58.88%	2 048
	10	60.42%	256
Standardization	5	55.93%	128
	10	58.59%	1 024

Table 4: Upsampling: The best results obtained with upsampling in cross-validation training.

our Hungarian emotion database is smaller and not a balanced one (57–61% of the dataset has the label "neutral"), we decided to use upsampling on our BoAW features before SVM learning. In this scenario, we tested how upsampling affects our results. We carried out tests with 5 and 10 neighbours, standardization and normalization, as previously and we utilized the k-means clustering algorithm.

Based on the results (see Table 4 and Figure 10), we can state that upsampling gave an improvement of about 10% compared to all of our previous results. In addition, perhaps the biggest advantage is that we were able to further reduce the optimal codebook size, which is good in terms of the speed and degree of difficulty of the learning process. It has only one disadvantage, namely our training curve is not as stable as before. There are two peaks here instead of one and it leads to less predictable learning. Because of this, we applied upsampling in our next experiment.

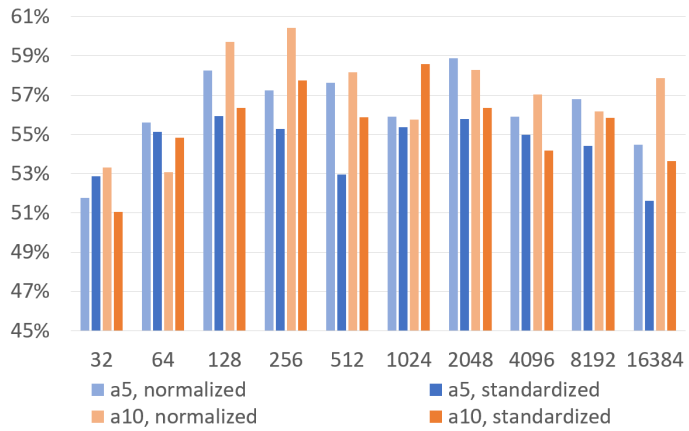

Figure 10: Upsampling: The results obtained with upsampling in cross-validation training.

Feature-preprocessing	a	Maximum UAR	Codebook size
Normalization	5	58.63%	512
	10	57.48%	512
Standardization	5	56.08%	512
	10	59.00%	512

Table 5: Deltas: The best results of cross-validation using deltas.

4.5 Derivatives

As we mentioned previously, using derivatives is a frequently used technique in speech processing to get information about the speaker's change of voice over time. In the last optimizing scenario we tested the effect of using these deltas. Our experiments so far have shown that the 16 384-sized codebooks always give a lower performance, and working with big dimensions slows down the training process. Because of the low performance we no longer need to run cross-validation and test with this huge 16 384 size. It should be added that the codebook sizes on Figure 11 had to be doubled, because we created two unit-sized codebooks; one for the original features and one for deltas and we used both of them while training.

From our results (see Table 5 and Figure 11), by using deltas we managed to reduce the number of necessary and sufficient codewords to a moderate size. Another advantage is that training trends are less random than before and much more predictable. So owing to this positive result, the final evaluation with the test database was performed with deltas.

4.6 Final tests

All of our previous decisions were made based on the optimal results got by the cross-validation performed on the teaching set, so our final set of BoAW parameters are the following:

- 5 neighbours, normalization, upsampling and using deltas
- 10 neighbours, normalization, upsampling and using deltas
- 5 neighbours, standardization, upsampling and using deltas
- 10 neighbours, standardization, upsampling and using deltas

All of our previous results indicate that above a codebook size of 64 the results display consistently increasing trends. Because of this, we decided to

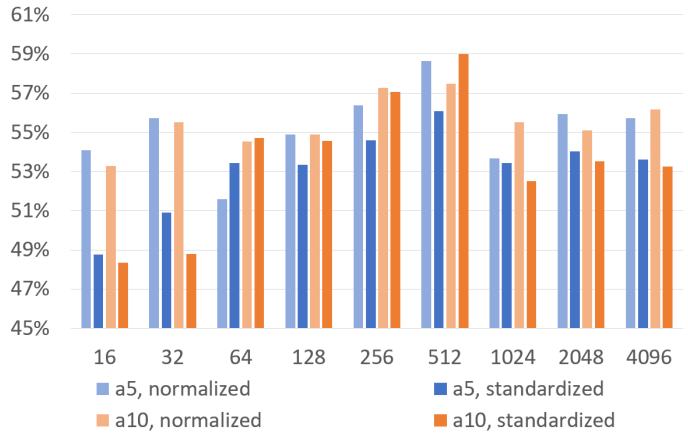

Figure 11: Deltas: The results of cross-validation using deltas.

Feature-preprocessing	**a**	**Maximum UAR**	**Codebook size**
Normalization	5	68.68%	64
	10	67.77%	128
Standardization	5	71.15%	64
	10	65.42%	64

Table 6: Final tests: The best results of the final tests without cross-validation.

run our tests with a size of 128 (the codebook size is 64 on the test diagrams, because we have to double the size when using deltas).

Based on our final results (see Table 6 and Figure 12), we may conclude that the bag-of-audio-words representation can be utilized for speech emotion recognition. It can be seen that with the right parameter settings we were able to reduce the dimension of the best result. However the trends of the test results are not clear, and the connection between increasing codeword quantities and decreasing evaluation results also seems to suggest that the larger the codebook size we choose, the greater the chance of over-fitting and our classifier will lose its ability to generalize.

The best results on the test set are close to 70%, which is at least as good as the other published paper results. Our final percentage scores cannot be compared directly to previous published results for this database because they used accuracy instead of UAR and had no upsampling. However we can still state that BoAW is as good representation as any other if we carefully optimize the parameter values of the algorithm.

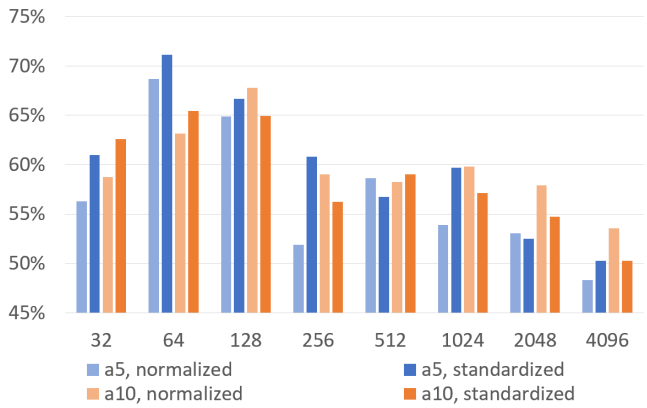

Figure 12: Final tests: The results of the final tests without cross-validation.

5 Conclusions

In this study, the bag-of-audio-words feature representation method was used for speech emotion recognition with a Hungarian emotional database. Because the BoAW method has many adjustable parameters, we had to train a lot of machine learning models with different parameter value combinations, so the training runtimes was an important consideration. Although each model building and evaluation did not take a long time, due to the many combinations and possible correction runs, the whole experiment took far too far.

From our results, we offer some useful suggestions that might be helpful when using openXBOW. These are:

- Transform the input sample set to the same scale by normalization or standardization. This is always a good choice.
- For greater generalization ability, it is worth including more neighbours in the quantizing step, such as 5 or 10.
- It is worth choosing the size of codebook from a medium-large range (e.g. between 128 and 4 096). If possible, try to keep the codebook size low to get a better generalization.
- Clustering the k-means and k-means++ algorithms are worth exploring.
- By balancing the frequency of classes seen during learning (in this case with upsampling), we can improve our generalization ability.
- We should calculate and use the deltas.

Now we see that the bag-of-audio-words technique is competitive in the area of speech emotion recognition, but this method has several parameter values that need to be precisely tuned for optimal efficiency.

There are several directions we can pursue in the future. Firstly, we would like to see if we can use other database codebooks to extract BoAW features from different databases. In another words, we wish to know whether codebooks are portable and what the best codebooks are for different purposes. We could also test other frame-level feature sets to see if there are any practical benefits of using them.

Acknowledgements

This study was partially funded by the National Research, Development and Innovation Office of Hungary via contract NKFIH FK-124413, by grant NKFIH-1279-2/2020 of the Hungarian Ministry of Innovation and Technology, and by the Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Programme. Gábor Gosztolya was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences, and by the ÚNKP-21-5 New National Excellence Program by the Hungarian Ministry of Innovation and Technology.

References

[1] D. Arthur, S. Vassilvitskii, k-means++: the advantages of careful seeding. *Proceedings of SODA*. **8** (2007) pp. 1027–1035, https://dl.acm.org/doi/10.5555/1283383.1283494 ⇒6

[2] S. Bernhard, C. John, S. John, J. Alexander, C. Robert, Estimating the support of a high-dimensional distribution. *Neural Computation* **13** (2001) 1443–1471, https://doi.org/10.1162/089976601750264965 ⇒8

[3] F. Burkhardt, M. Ballegooy, K. Engelbrecht, T. Polzehl, J. Stegmann, Emotion detection in dialog systems: Applications, strategies and challenges, *Proceedings Of ACII*. 2009, pp. 985–989. ⇒2

[4] C. Chang, C. Lin, LIBSVM: A library for support vector machines, *ACM Transactions On Intelligent Systems And Technology* **2**, 3 (2011) 1–27, https://doi.org/10.1145/1961189.1961199 ⇒8

[5] G. Csurka, F. Perronnin, Fisher Vectors: Beyond bag-of-visual-words image representations, *Computer Vision, Imaging And Computer Graphics, Theory And Applications* **229** (2011) 28–42, http://dx.doi.org/10.1007/978-3-642-25382-9_2 ⇒3

[6] L. Deng, J. Droppo, A. Acero, A Bayesian approach to speech feature enhancement using the dynamic cepstral prior, *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing* **1** (2002) 829–832, http://dx.doi.org/10.1109/ICASSP.2002.5743867 ⇒6

[7] F. Eyben, M. Wöllmer, B. Schuller, Opensmile: The Munich Versatile and Fast Open-source Audio Feature Extractor. *Proceedings Of ACM Multimedia*. (2010) pp. 1459–1462, http://doi.acm.org/10.1145/1873951.1874246 ⇒7

[8] G. Gosztolya, Using the Fisher vector representation for audio-based emotion recognition. *Acta Polytechnica Hungarica*. **17** (2020) 7–23, http://dx.doi.org/10.12700/APH.17.6.2020.6.1 ⇒3, 7

[9] T. Grósz, L. Tóth, A comparison of deep neural network training methods for large vocabulary speech recognition. *Proceedings Of TSD*, 2013, pp. 36–43, http://dx.doi.org/10.1007/978-3-642-40585-3_6 ⇒3

[10] M. Hossain, G. Muhammad, Cloud-assisted speech and face recognition framework for health monitoring. *Mobile Networks And Applications*. **20** (2015) 391–399, https://doi.org/10.1007/s11036-015-0586-3 ⇒2

[11] D. Issa, M. F. Demirci, A. Yazici, Speech emotion recognition with deep convolutional neural networks, *Biomedical Signal Processing and Control*, **59** (2020) 101894, http://dx.doi.org/10.1016/j.bspc.2020.101894 ⇒3

[12] J. James, L. Tian, C. Inez Watson, An open source emotional speech corpus for human robot interaction applications, *Proceedings Of Interspeech* (2018) pp. 2768–2772, https://doi.org/10.21437/Interspeech.2018-1349 ⇒2

[13] C. Jones, J. Sutherland, Acoustic emotion recognition for affective computer gaming. **1** (2008), https://doi.org/10.1007/978-3-540-85099-1 ⇒2

[14] F. Metze, A. Batliner, F. Eyben, T. Polzehl, B. Schuller, S. Steidl, Emotion recognition using imperfect speech recognition. *Proceedings Of Interspeech*, 2010, pp. 478–481, http://dx.doi.org/10.21437/Interspeech.2010-202 ⇒8

[15] S. Pancoast, M. Akbacak, Bag-of-audio-words approach for multimedia event classification. *Proceedings Of Interspeech*, 2012, pp. 2105–2108. ⇒3

[16] S. Pancoast, M. Akbacak, Softening quantization in bag-of-audio-words. *Proceedings Of ICASSP*, 2014, pp. 1370–1374, https://doi.org/10.1109/ICASSP.2014.6853821 ⇒6

[17] J. Přibil, A. Přibilová, J. Matoušek, GMM-based speaker age and gender classification in Czech and Slovak. *Journal Of Electrical Engineering*, **68** (2017) 3–12, http://dx.doi.org/10.1515/jee-2017-0001 ⇒2

[18] P. Raghavendra, W. Tianzi, V. Jesus, C. Nanxin, D. Najim, X-vectors meet emotions: A study on dependencies between emotion and speaker recognition, *ICASSP 2020 – 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7169–7173, http://dx.doi.org/10.1109/ICASSP40776.2020.9054317 ⇒3

[19] S. Rawat, P. Schulam, S. Burger, D. Ding, Y. Wang, F. Metze, Robust audio-codebooks for large-scale event detection in consumer videos, *Proceedings Of Interspeech*, 2013, pp. 2929–2933 ⇒6

[20] M. Schmitt, F. Ringeval, B. Schuller, At the border of acoustics and linguistics: Bag-of-audio-words for the cecognition of emotions in speech, *Proceedings Of Interspeech*, 2016, pp. 495-499, http://dx.doi.org/10.21437/Interspeech.2016-1124 ⇒3

[21] M. Schmitt, B. Schuller, openXBOW – Introducing the Passau open-source crossmodal bag-of-words toolkit. *Journal Of Machine Learning Research* **18** (2017) 1–5, http://jmlr.org/papers/v18/17-113.html ⇒5

[22] B. Schuller, S. Steidl, A. Batliner, A. Vinciarelli, K. Scherer, F. Ringeval, et al., The INTERSPEECH 2013 computational paralinguistics challenge: Social signals, conflict, emotion, autism, *Proceedings Of Interspeech*, 2013, pp. 148–152. ⇒7

[23] D. Sztahó, V. Imre, K. Vicsi, Automatic classification of emotions in spontaneous speech, *Proceedings Of COST 2102*, 2011, pp. 229–239, http://dx.doi.org/10.1007/978-3-642-25775-9_23 ⇒3, 7

[24] M. Swain, A. Routray, P. Kabisatpathy, Databases, features and classifiers for speech emotion recognition: a review, *Int J Speech Technol* **21** (2018) 93–120, https://link.springer.com/article/10.1007/s10772-018-9491-z ⇒3

[25] K. Vicsi, D. Sztahó, Recognition of emotions on the basis of different levels of speech segments. *Journal Of Advanced Computational Intelligence And Intelligent Informatics* **16** (2012) 335–340, http://dx.doi.org/10.20965/jaciii.2012.p0335 ⇒3, 7

[26] L. Vidrascu, L. Devillers, Detection of real-life emotions in call centers, *Proceedings Of Interspeech*, 2005, pp. 1841–1844, http://dx.doi.org/10.21437/Interspeech.2005-582 ⇒2

[27] T. Zhang, J. Wu, Speech emotion recognition with i-vector feature and RNN model, *2015 IEEE China Summit and International Conference on Signal and Information Processing (ChinaSIP)*, 2015, pp. 524-528, http://dx.doi.org/10.1109/ChinaSIP.2015.7230458 ⇒3

[28] Y. Zhang, R. Jin, Z. Zhou, Understanding bag-of-words model: A statistical framework. *International Journal Of Machine Learning And Cybernetics* **1** (2010) 43–52, http://dx.doi.org/10.1007/s13042-010-0001-0 ⇒3

Received: November 23, 2021 • Revised: February 1, 2022