This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAFFC.2018.2871450, IEEE Transactions on Affective Computing



# Social Signal Detection by Probabilistic Sampling DNN Training

Gábor Gosztolya, Tamás Grósz, and László Tóth, *Member, IEEE*

**Abstract**—When our task is to detect social signals such as laughter and filler events in an audio recording, the most straightforward way is to apply a Hidden Markov Model – or a Hidden Markov Model/Deep Neural Network (HMM/DNN) hybrid, which is considered state-of-the-art nowadays. In this hybrid model, the DNN component is trained on frame-level samples of the classes we are looking for. In such event detection tasks, however, the training labels are seriously imbalanced, as typically only a small fraction of the training data corresponds to these social signals, while the bulk of the utterances consists of speech segments or silence. A strong imbalance of the training classes is known to cause difficulties during DNN training. To alleviate these problems, here we apply the technique called probabilistic sampling, which seeks to balance the class distribution. Probabilistic sampling is a mathematically well-founded combination of upsampling and downsampling, which was found to outperform both of these simple resampling approaches. With this strategy, we managed to achieve a 7–8% relative error reduction both at the segment level and frame level, and we efficiently reduced the DNN training times as well.

**Index Terms**—Deep neural networks, instance sampling, social signals, laughter detection.

## 1 Introduction

Within speech technology, an emerging area is paralinguistic phenomenon detection, which seeks to locate non-linguistic events (conflict, laughter events, etc.) in speech. One task belonging to this area is the detection of social signals, from which perhaps laughter and filler events (vocalizations like "um", "eh", "er" etc.) are the most important. Many experiments were performed with the goal of detecting laughter (e.g. [1], [2], [3]), and the detection of filler events has also become popular recently (e.g. [4], [5], [6], [7], [8], [9]).

Most of the earlier studies focused on the frame level (e.g. [5], [10], [11]). However, a more realistic approach is to detect occurrences of the given phenomena. To do this, one may simply borrow techniques from Automatic Speech Recognition (ASR), such as applying a Hidden Markov Model (HMM) to combine the local (frame-level) likelihood estimates of a Gaussian Mixture Model (GMM) into segment-level occurrence hypotheses. Nowadays in ASR, with the invention of Deep Neural Networks (DNNs), HMM/GMMs have been replaced by the so-called HMM/DNN hybrids as state-of-the-art [12], and this technique may be readily applied in this task as well.

To construct a HMM/DNN-based event detector, we train our DNNs on frame-level samples of the classes we are looking for (e.g. laughter) and background events (e.g. speech, silence, background noise). However, social signals are relatively rare phenomena in spontaneous speech, typically taking up only a small fraction of speech time [13]. This means that we have to train our DNNs on a data set where the distribution of examples belonging to the different classes are heavily imbalanced. Neural networks are known to be sensitive to (a strong) class imbalance (see e.g. [14]), and DNNs are no exception. Fortunately, in ASR the distribution of the phones are not so imbalanced that it would cause noticeable training difficulties. Also, we usually apply context-dependent state tying (see e.g. [15], [16], [17]), which also tends to reduce the difference in the number of examples belonging to the different states (i.e. classes). However, in the case of social signal detection the imbalance of the classes is much more dramatic, and it needs to be handled carefully.

The simplest solution to help balance the distribution of training samples associated with the different classes is to re-sample the input data. In its simplest form it means that we simply discard training data from the more frequent classes (*filtering* or *down-sampling* approach [18]), which clearly decreases the variability of the training data, and this may result in a loss of accuracy.

Of course, re-sampling is not the only way to balance the class distribution of the training data. Another choice, being very popular in image processing, is that of *data augmentation*; that is, generating further training examples from the existing ones. In image processing, this typically means rotating, mirroring and resizing images, and changes in the colours [19], [20], [21]. In ASR, some studies suggest that speeding up or slowing down utterances slightly [22], adding noise to the speech signal [23] or utilizing vocal tract length perturbation [24], [25] are useful methods to increase the size of training sets in low-resource scenarios. Notice, however, that while generating new images can easily be applied for balancing class distribution, in speech processing, where each utterance tends to contain thousands of training examples associated with several classes, it is not so straightforward to realize. The ASR studies listed above all focused on creating more training examples in general,

- *G. Gosztolya and L. Tóth are with the MTA-SZTE Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and the University of Szeged, Hungary.*
  *E-mail: { ggabor, tothl } @ inf.u-szeged.hu*
- *T. Grósz is with the University of Szeged, Hungary.*
  *E-mail: groszt@inf.u-szeged.hu*

*Manuscript received April 19, 2005; revised August 26, 2015.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAFFC.2018.2871450, IEEE Transactions on Affective Computing



and not just for specific classes.

The data augmentation approaches mentioned so far all operate *before* feature extraction, i.e. they create "new" images or speech recordings. Another type of data augmentation methods operate in the "feature space" instead of this "data space": they create synthetic examples based on the *feature vectors* of existing training examples. Perhaps the most popular such algorithm is the Synthetic Minority Over-sampling Technique (SMOTE, [26]), which takes the linear combination of $k$ nearest neighbours of a randomly chosen example in the feature space.

These methods, despite being more task-independent than the former ones, appear to be quite hard to apply in ASR (or in the detection of laughter or filler events). The reason for this is that it is common practice to train the acoustic DNNs on a sliding window of feature vectors to improve performance. As the consecutive feature vectors of an utterance are typically stored after each other in the memory, using the feature vectors of the neighbouring frames imposes no additional memory requirement. However, if we generate further synthetic examples via SMOTE (or some similar method), we have to handle the neighbouring feature vectors as well. If we leave them as they are, our newly generated examples will differ from the original ones only in a fraction of their feature values. However, to use SMOTE for generating new feature values for the "neighbouring" frames as well would expand our memory requirements (e.g. using 7 frames at each side leads us to use 15 times as much memory), since the generated "neighbours" cannot be re-used for other training examples.

For these reasons, in this study, to detect laughter and filler events in English mobile phone conversations, we will not apply data augmentation, but focus on training data re-sampling. Instead of the simple approaches of down- and upsampling (i.e. use the examples of the rarer classes less and more frequently, respectively), we will utilize a more sophisticated technique called *probabilistic sampling*, which is a mathematically well-founded combination of the two approaches. We will also present an efficient way of implementing this sampling strategy, which meets the needs of speech processing.

The structure of this paper is as follows. First, we will introduce probabilistic sampling, give an implementation approach which efficiently reduces the randomness present during DNN training, and discuss how applying probabilistic sampling affects the application of DNNs in Hidden Markov Models. Then we will describe the experimental set-up: the database used, the DNN parameters, the language model used, explain the way the meta-parameters were fine-tuned, and discuss the evaluation metrics used. Afterwards we will present and analyze our results, where besides event detection accuracy, we examine how the application of probabilistic sampling affected the training times of our DNN-based acoustic models, and also look at how the output likelihood values changed.

## 2 Probabilistic Sampling

Like most machine learning algorithms, neural nets are sensitive to class imbalances, and tend to behave inaccurately on classes having only a few examples. The simplest solution to help balance the class distribution is to downsample the more frequent classes, but it results in data loss, and hence may also result in a drop in accuracy. A more refined solution is to *upsample* the rarer classes: we utilize the examples from these classes more frequently during training. A mathematically well-formulated re-sampling strategy is the training scheme called probabilistic sampling [27], [28]. This procedure selects the next training sample following a two-step sampling scheme: first we select a class according to some probability distribution, then we randomly pick a training sample that belongs to this class. For the first step, we assign the following probability to each class:

$$P(c_k) = \lambda \frac{1}{K} + (1 - \lambda) Prior(c_k), \tag{1}$$

where $Prior(c_k)$ is the prior probability of class $c_k$, K is the number of classes and $\lambda \in [0, 1]$ is a parameter. For $\lambda = 0$, the above formula returns the original class distribution, so probabilistic sampling will behave just as conventional sampling does. For $\lambda = 1$, we get a uniform distribution over the classes, so we get totally balanced samples with respect to class frequency. Selecting a $\lambda$ value between 0 and 1 allows us to interpolate between the two distributions. (For the sake of simplicity, for each iteration we select the same number of samples as there are in the training set.)

Database sampling is quite rarely applied either in ASR or in similar areas. We should mention the in-depth study of García-Moral et al. [29], who discarded examples belonging to the more common classes. Although this made the ANN training process much faster, they also experienced a slight drop in accuracy. Tóth and Kocsor applied this probabilistic sampling approach to a very small speech recognition task in 2005 [28], and they were able to improve the recognition accuracy. More recently, Grósz et al. applied probabilistic sampling in the training of DNN acoustic models with context-dependent targets. With this strategy they obtained significant reductions in the word error rate in two large-vocabulary speech recognition tasks [30]. These studies focused on speech recognition, where the class imbalance is relatively small; still the probabilistic sampling technique resulted in performance gains. When the task is social signal detection, where the class imbalance is usually much larger, we can reasonably expect that applying this technique will result in significant improvements in the scores.

### 2.1 Efficient Implementation for Speech Processing

Notice that the basic version of the probabilistic sampling scheme introduces randomness in the sampling process at two distinct points: first we *randomly* pick a class, and then we *randomly* pick a training sample of the given class. Although usually we have millions of training frames, for some events of interest the number of training examples is much more limited. If we follow this scheme, it may happen that some examples are not used at all during training.

In standard ANN training, the typical data randomization process consists of shuffling the data vectors in a random order before training. This process guarantees that all the training samples get processed. This gave us the idea of implementing probabilistic sampling as follows.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAFFC.2018.2871450, IEEE Transactions on Affective Computing



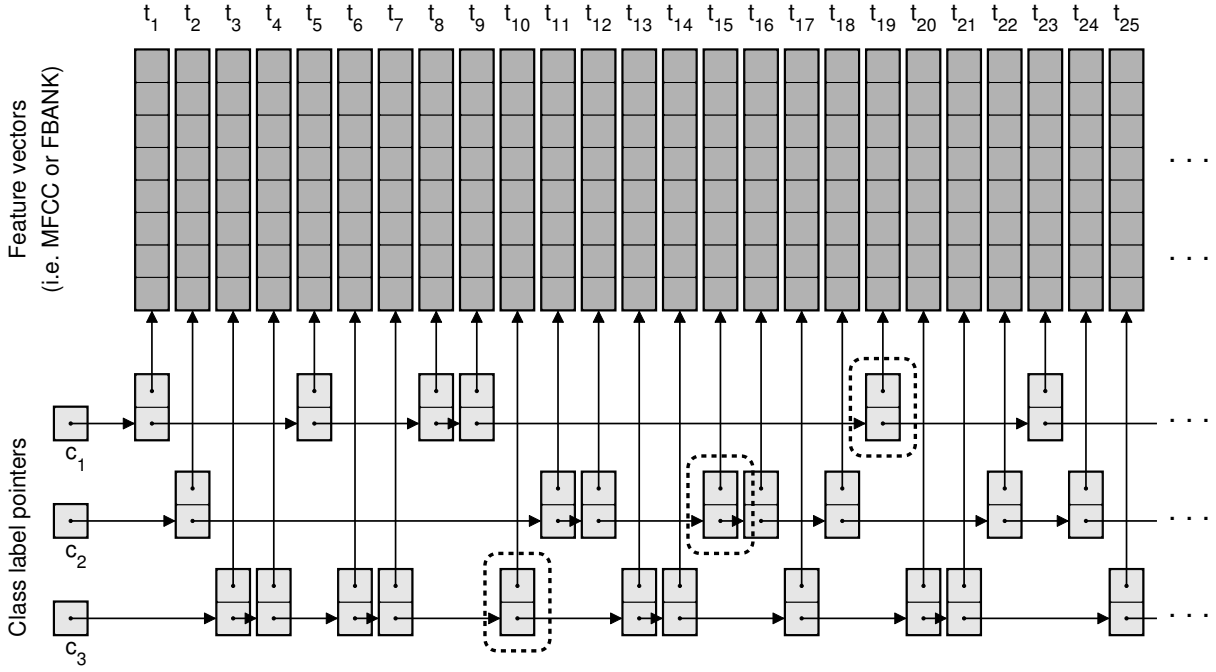


Fig. 1. The basic scheme of the proposed probabilistic sampling implementation approach. The pointers to the frame-level feature vectors are stored in a linked list data structure separately for each class, allowing the efficient location of the next training sample of each class and easy access to the neighbouring frame-level feature vectors. The dashed circles show the pointers to the next training instance for each class employed during training.

In our approach, first we set up separate linked list data structures [31] for each class. These linked lists store pointers to each frame-level feature vector associated with the given class (see Figure 1). Then we shuffle the nodes of each linked list (i.e. the training examples). Before starting the first training iteration, we also set pointers to the first example of *each class*.

During training, first we randomly select a class, following the multinomial distribution of Eq. (1). Then, however, instead of randomly picking a sample within the class, we use the training sample (i.e. frame) indicated by the pointer of the given class, and advance the pointer to the next sample. (If this was the last sample of the actual class, we set our pointer to the first one.) This way we can guarantee that, for each class, all of its training samples will be used with roughly the same frequency, avoiding the possibility of not using or underusing certain samples. Notice that with this approach we need only two pointers for each example (one pointing to the address of the actual feature vector, and one referring to the next node of the linked list), which is negligible, compared to the size of the training data.

Another important aspect of various frame-level speech tasks is that nowadays it is standard practice to train frame-level classifiers on a *sliding window* of neighbouring frames instead of using only the feature vector of a single frame. That is, instead of using the $x_t$ observation vector associated with the $t$th frame, we use the $2k+1$-frame long $x_{t-k}, \ldots, x_t, \ldots, x_{t+k}$ concatenated feature vector to classify the $t$th frame. If we had opted for simply sorting the feature vectors by their class labels, we would have had to store the feature values belonging to the neighbouring frames as well, which would multiply the memory requirements of DNN training. However, by storing only a pointer referring to the location of the feature vector of the given frame, we can easily access the feature vectors of both the preceding and the subsequent neighbouring frames.

### 2.2 Application in a HMM/ANN Hybrid

A standard Hidden Markov Model requires frame-level estimates of the class-conditional likelihood $p(x_t|c_k)$ for the given observation vector $x_t$, which are provided by a generative method like Gaussian Mixture Models (GMMs) [32]. When we replace GMMs by neural networks, which estimate $P(c_k|x_t)$, the $p(x_t|c_k)$ values expected by the HMM can be got using Bayes' theorem. So, in a HMM/ANN hybrid, we divide the posterior estimates produced by the ANN (or DNN) by the priors of the classes, i.e. by $Prior(c_k)$. This will give the required likelihood values within a scaling factor, which can be ignored as it has no influence on the subsequent search process.

In contrast, Tóth and Kocsor showed that when we train the neural network using a uniform class distribution (being equivalent to using probabilistic sampling with $\lambda = 1$), the networks will estimate directly the class-conditional $p(x_t|c_k)$ values within a scaling factor [28]. This means that when we integrate our classifiers trained with $\lambda = 1$ into a HMM/ANN hybrid, we should omit the division of the network outputs by the class priors.

Theoretically we should either use $\lambda = 0$ and divide by the class priors, or use $\lambda = 1$ and not divide by the priors. However, in practice the probability estimates are not precise. While with standard sampling ($\lambda = 0$) ANNs tend to underestimate the probability of the rarer classes, chances are that they will be overestimated in the case of

TABLE 1
The distribution of laughter and filler events in the SSPNet Vocalization Corpus

| Set | Total Count | | | Total Duration (sec) | | | Total Duration (%) | |
|---|---|---|---|---|---|---|---|---|
| | Utterances | Laughter | Fillers | Utterances | Laughter | Fillers | Laughter | Fillers |
| Training | 1583 | 649 | 1710 | 17358 | 593 | 850 | 3.4% | 4.9% |
| Development | 500 | 225 | 556 | 5478 | 258 | 294 | 4.7% | 5.4% |
| Test | 680 | 284 | 722 | 7444 | 240 | 355 | 3.2% | 4.8% |
| Total | 2763 | 1158 | 2988 | 30280 | 1091 | 1499 | 3.6% | 5.0% |

uniform sampling ($\lambda = 1$). Tóth and Kocsor [28] achieved the best performance with $\lambda$ values strictly between 0 and 1, and we also recommend finding the best value for the given task experimentally.

In this study we propose another likelihood normalization strategy. The explanation presented in [28] tells us why (theoretically) we have to divide the output likelihoods by the original class priors when we train DNNs with $\lambda = 0$, and why we should avoid any division in the case of $\lambda = 1$. Experimentally, however, intermediate $\lambda$ values were found to be optimal. With these intermediate $\lambda$ values, actually a *third* class distribution is used during training. This means that we should in fact divide the output likelihood estimates of a DNN by the prior probabilities of the classes given by the $P(c_k)$ values calculated using Eq. (1). As the third option, we also tested this approach for both $\lambda = 0$ and $\lambda = 1$, as well as for intermediate $\lambda$ values.

## 3 Experimental Setup

### 3.1 The SSPNet Vocalization Corpus

We performed our experiments on the SSPNet Vocalization Corpus [4], which consists of 2763 short audio clips extracted from telephone conversations, containing 2988 laughter and 1158 filler events (see Table 1). In this corpus just $3.6\%$ of the duration corresponds to laughter, and $5.0\%$ consists of filler events; the remaining $91.6\%$ consists of miscellaneous speech ($51.2\%$) and silence ($40.2\%$). Unfortunately, in the public annotation only the laughter and filler events are marked, so in our experiments each frame was labeled as one of three classes: "laughter", "filler" or "garbage" (meaning both silence and non-filler non-laughter speech). We will follow the standard split of the dataset into a training, development and test set, introduced at the Interspeech Computational Paralinguistics Challenge (ComParE) in 2013 [10]. From the total of 2763 clips, 1583 were assigned to the training set, 500 clips to the development set, and 680 clips to the test set.

### 3.2 Evaluation Metrics

For tasks like social signal detection, where the distribution of classes is significantly biased, classification accuracy is only of limited use. Also, although on this dataset and in social signal detection in general it is common to rely on the Area-Under-the-Curve (AUC) score of the frame-level posteriors for the classes of interest (now laughter and filler events) (e.g. [5], [10], [33], [34]), it was shown recently (see [35]) that frame-level AUC is an unreliable metric for this task. Besides raising several theoretical objections, Gosztolya showed experimentally that the AUC values can be significantly improved by applying a simple smoothing over time on the output likelihoods (a technique applied in several studies, e.g. [5], [11], [33], [34], [36], [37]), but this transformation makes the scores unsuitable for a Hidden Markov Model.

Because of this, we decided to apply a HMM to perform event occurrence detection, and calculated accuracy metrics based on these occurrences. We applied the standard information retrieval metrics of precision, recall and F-measure (or $F_1$-score) [38]. We combined the scores of the two social signals by *macro-averaging*: we averaged the precision and recall scores of the two phenomena in an unweighted manner, and calculated $F_1$ from these average values.

Another open question is how we should calculate the number of true positives, false positives and false negatives. One way to do this is that, *after applying the Hidden Markov Model*, we compare the resulting event occurrence hypotheses with the manual annotation frame-by-frame. This approach was followed by e.g. Salamin et al. [4]. There are similar evaluation approaches available for the task of speaker diarization as well (see e.g. [39], [40]).

However, frames are in fact an intermediate step required only for technical reasons, and they are used relatively rarely in actual evaluations. In ASR only the phoneme (word) sequence recognized is compared with the ground truth transcript, and the time-alignment is completely ignored. The reason for this is partly that it is impossible to objectively position phoneme boundaries within a 10ms precision, which is the typical frame-shift: due to the continuous movement of the vocal chords and the mouth, we can expect no clear-cut boundary between two consecutive phonemes. Even from the aspect of user expectations, locating an event occurrence with slight differences at the starting and ending positions still obviously counts as a perfect match. The requirement of frame-level precision is avoided if we rate the performance at the segment level, where slight differences in time alignment are tolerated.

To decide whether two occurrences of events (i.e. a laughter occurrence hypothesis returned by the HMM and one labelled by a human annotator) match, there is no de facto standard in the literature. For example, Gosztolya required only that the two occurrences refer to the same kind of event (i.e. in our case laughter or filler) and that their time intervals intersect [35]. Pokorny et al. [41] also matched the occurrences by checking whether their time intervals intersected, allowing several event occurrence hypotheses to match one manually annotated occurrence. In the NIST standard for Spoken Term Detection evaluation [42], the centre of the two occurrences have to fall close to each other

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAFFC.2018.2871450, IEEE Transactions on Affective Computing



(i.e. within a threshold). In this study we combined the two approaches: we required that the two occurrences intersect, while their centre also had to be close to each other (within 500ms, as in the NIST STD standard [42]).

We performed our experiments by measuring $F_1$ both at the segment level and at the frame level. As the optimal meta-parameters (language model weight and $\lambda$) may differ for the two (evaluation) approaches, we found them independently for the two kinds of metrics applied.

### 3.3 HMM State Transition Probabilities

Our HMM/DNN hybrid set-up for vocalization occurrence detection consisted of only three states, each one representing a different acoustic event (i.e. laughter, filler and garbage). In this set-up, the state transition probabilities of the HMM practically correspond to a language model. Following the study of Salamin et al. [4], we constructed a frame-level state bi-gram, calculated from statistics of the training set. The probability values of the transitions were calculated on the training set, while we used the development set to find the optimal language model weight. As the last step, evaluation on the test set was performed using this language model weight.

### 3.4 DNN Parameters

We employed DNNs with 5 hidden layers, each containing 256 rectified neurons, based on the results of preliminary tests, while we applied the softmax function in the output layer. We used our custom DNN implementation, which achieved the best accuracy known to us on the TIMIT database with a phonetic error rate of $16.5\%$ on the core test set [43]. We tested three feature sets; the first one was the standard 39-sized MFCC + $\Delta$ + $\Delta\Delta$ feature set, frequently used both in phoneme classification (e.g. [44], [45]) and in laughter detection (e.g. [1], [46], [47]). As DNNs tend to perform better on more primitive features (see e.g. [12]), we also calculated 40 raw mel filter bank energies along with energy and their first and second order derivatives (123 values overall; *FBANK* feature set). Both sets were extracted using the HTK tool [48]. The third feature set was the one provided for the ComParE 2013 Challenge [10]. It consisted of the 39-sized MFCC + $\Delta$ + $\Delta\Delta$ feature vector with voicing probability, HNR, $F_0$ and zero-crossing rate, and their derivatives. To these 47 features their mean and standard derivative in a 9-frame neighbourhood were added, resulting in a total of 141 features [10]. Previous studies (e.g. [11], [33], [36]) found this feature set to be quite useful for detecting laughter and filler events; we extracted these attributes with the openSMILE tool [49].

We performed our experiments on a PC with Intel i5 CPU operating at 3.2 GHz and having 8GB of RAM. To speed up DNN training, we opted for using a GPU (a GeForce GTX 570 device from the NVIDIA Corporation), which is standard practice nowadays.

It is known that DNN training is a stochastic procedure due to random weight initialization. Probabilistic sampling introduces a further random factor by randomly choosing the class of the next training example following the multinomial distribution in Eq. (1). To counter this effect, for each meta-parameter setting (feature set, and the $\lambda$ value for probabilistic sampling) we trained five DNN models; we primarily rated the tested methods on their *average* performance, but we also list the best and worst scores.

### 3.5 Probabilistic Sampling

We evaluated the probabilistic sampling technique by varying the value of $\lambda$ in the range $[0, 1]$ with a step size of $0.1$. We will list the accuracy scores measured on the test set, but we chose the value of $\lambda$ based on the results obtained on the development set. As we cannot know in advance whether we should divide the DNN output likelihoods by the class priors, avoid this re-scaling, or use the formula of Eq. (1) as priors, we decided to test all three approaches for all $\lambda$ values. Note that in order to do this we did not have to train any additional DNNs, as this transformation affects only the output likelihood scores.

## 4 Results

### 4.1 Baseline scores

We treated standard sampling (i.e. using each training sample once in each epoch) as our baseline; however, first we have to determine the optimal number of neighbouring frame-level vectors used. We found the optimal parameter value by grid search; we tested sliding windows that had lengths of 1, 5, 9, ..., 65 frames. Since these DNNs were trained using all instances once in each epoch (i.e. $\lambda = 0$), the output likelihood scores were divided by the original class priors before utilizing them in the Hidden Markov Model. The utility of this transformation was also reinforced by our preliminary tests.

The minimum, maximum, and mean $F_1$ scores obtained this way are shown in Fig. 2. It can be seen that concatenating the feature vectors of the neighbouring frames indeed helps classification. We chose the sliding windows with a length of 21, 29 and 33, MFCC, FBANK and ComParE feature sets, respectively. Lowest mean scores were achieved via MFCC, while models trained with FBANK and ComParE had roughly the same performance, ComParE producing slightly higher average values. Taking the minimum and maximum values into consideration as well, we can see that the variation of scores is practically independent of the number of neighbours used or the feature set. In general, however, the segment-level $F_1$ scores tend to vary more than the frame-level scores; in our opinion this is because the key difference between the models trained is reflected in the detection or miss of shorter occurrences. These appear in the frame-level performance values only slightly, but they affect the segment-level $F_1$ scores more, since at this level an occurrence lasting only a couple of frames is just as important as one being over one second long (which, in our experience, is not unrealistic for a laughter event).

Table 2 contains the precision, recall and F-measure scores measured for the best sliding window sizes (21, 29 and 33, MFCC, FBANK and ComParE feature sets, respectively). The frame-level scores are higher than those reported by Salamin et al. [4], which is probably because we used DNNs for frame-level likelihood estimation, which can be considered as state-of-the-art, while Salamin et al. used Gaussian Processes. (Note that, despite the fact that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAFFC.2018.2871450, IEEE Transactions on Affective Computing



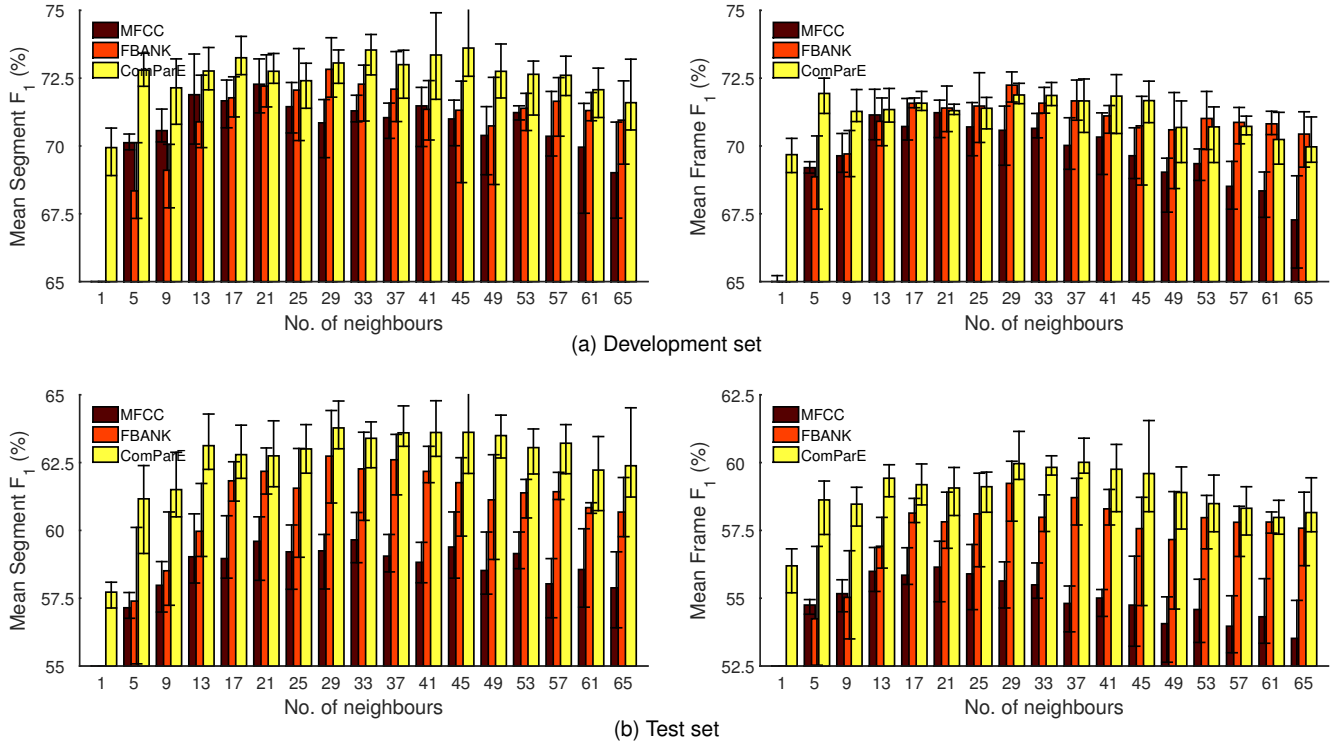


Fig. 2. Segment-level and frame-level macro-averaged $F_1$-scores obtained using standard backpropagation DNN training on the three feature sets applied, as a function of the number of neighbouring frame vectors used during training. The error bars denote minimum and maximum $F_1$ scores.

TABLE 2
Optimal number of neighbouring frames used during training and the corresponding segment- and frame-level precision, recall and $F_1$ scores for both the development and test sets (which will serve as our baseline)

| Evaluation | Data Subset | Feature Set | No. of frames | Laughter | | | Filler | | | Combined |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Prec. | Rec. | $F_1$ | Prec. | Rec. | $F_1$ | $F_1$ |
| Segment-level | Dev. | MFCC | 21 | 68.8% | 64.7% | 66.6% | 80.1% | 75.9% | 77.9% | 72.3% |
| | | FBANK | 29 | 75.6% | 63.4% | 69.0% | 82.0% | 72.0% | 76.7% | 72.8% |
| | | ComParE | 33 | 70.6% | 69.3% | 69.9% | 81.6% | 73.2% | 77.1% | 73.5% |
| | Test | MFCC | 21 | 49.2% | 57.8% | 53.1% | 65.8% | 66.0% | 65.9% | 59.6% |
| | | FBANK | 29 | 62.9% | 57.3% | 60.0% | 69.2% | 62.2% | 65.5% | 62.7% |
| | | ComParE | 33 | 58.0% | 62.4% | 60.1% | 66.3% | 67.1% | 66.7% | 63.4% |
| Frame-level | Dev. | MFCC | 21 | 67.8% | 74.8% | 71.1% | 67.5% | 75.6% | 71.3% | 71.2% |
| | | FBANK | 29 | 69.5% | 78.7% | 73.8% | 66.6% | 75.3% | 70.7% | 72.2% |
| | | ComParE | 33 | 70.2% | 76.3% | 73.0% | 70.7% | 70.5% | 70.6% | 71.9% |
| | | HMM/GPR [4] | — | 65.0% | 66.0% | 65.0% | 49.0% | 73.0% | 59.0% | 62.6% |
| | Test | MFCC | 21 | 43.7% | 70.9% | 54.0% | 52.8% | 63.3% | 57.6% | 56.1% |
| | | FBANK | 29 | 51.9% | 72.3% | 60.4% | 53.8% | 62.4% | 57.8% | 59.2% |
| | | ComParE | 33 | 54.6% | 70.4% | 61.4% | 54.3% | 62.6% | 58.1% | 59.8% |

the SSPNet Vocalization corpus is freely available, we found no other study that employed a Hidden Markov Model on it.) Overall, the accuracy scores obtained for laughter events are quite similar to those got for filler events. At the segment level we can see that precision is usually higher than recall, while we got the opposite at the frame level. We can also see that the segment-level scores obtained are higher for filler events than those for laughter, while this is not so when we compute the accuracy metrics at the frame level. The reason for this is probably that laughter occurrences are usually much longer than filler events. Indeed, in this corpus, laughter occurrences have an average duration of 942ms, while this is only 502ms for fillers (see Table 1). This difference means that there are three times as many filler event occurrences (segments) as there are laughter events. However, the number of laughter and filler *frames* do not differ to such a high extent, and this may lead to a more balanced performance at the frame level.

### 4.2 Results With Probabilistic Sampling

The average $F_1$ scores achieved by probabilistic sampling on the development set can be seen in Fig. 3; baseline values

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAFFC.2018.2871450, IEEE Transactions on Affective Computing



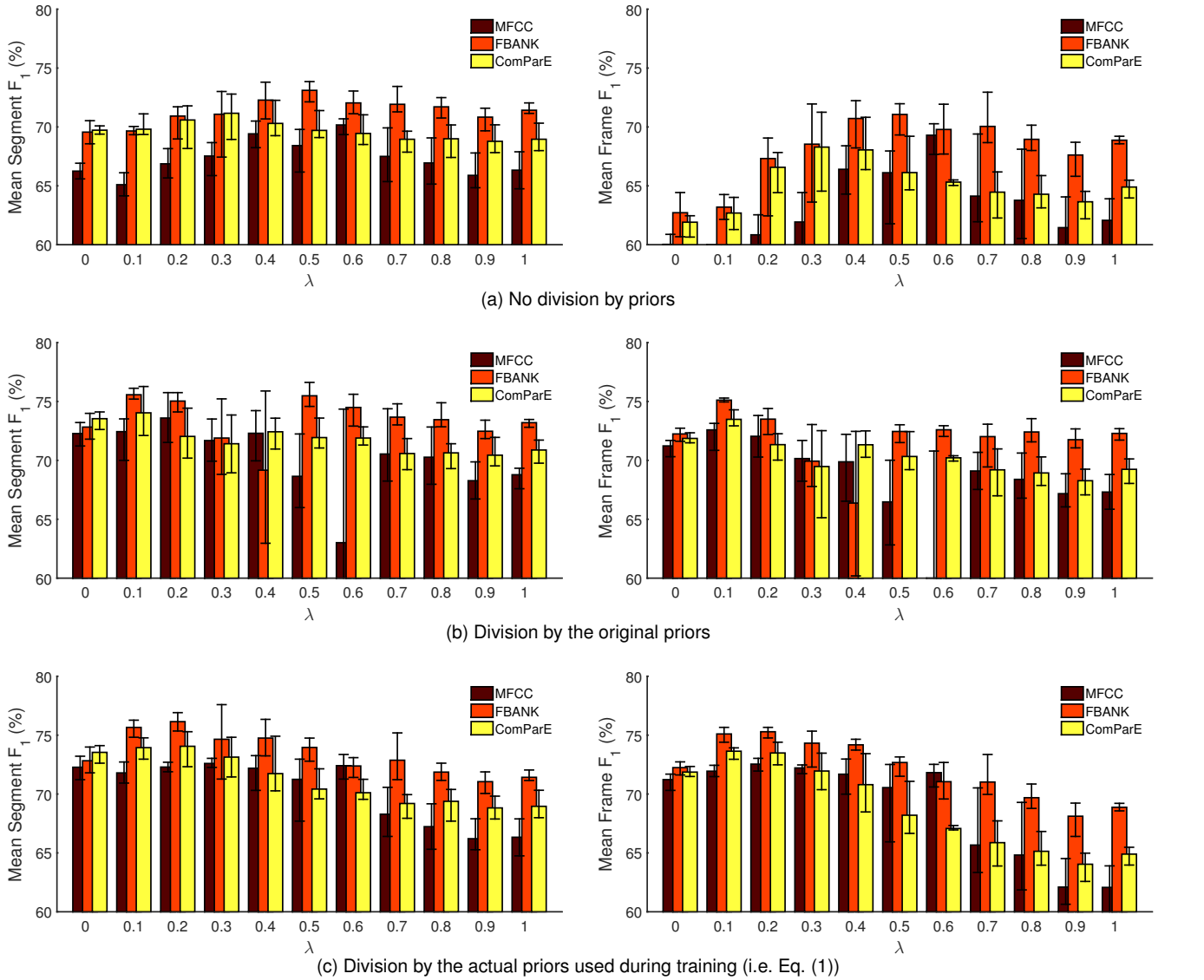


Fig. 3. Mean macro-averaged $F_1$ scores as a function of $\lambda$ on the development set, at the segment level (left) and frame level (right), when omitting division by the class priors (top), performing division by the original class priors (middle) and by the actual class priors used during training (bottom).

are shown as $\lambda = 0$. We can see that models trained on the MFCC feature set performed the worst. Regarding the FBANK and ComParE feature sets, while with the baseline setting (i.e. using full instance sampling) ComParE turned out to be the better one, when we applied probabilistic sampling we got our best scores usually by relying on the FBANK feature vectors. The scores obtained for the latter two feature sets are quite similar, though, so in our opinion any of these two could be used for efficient vocalization localization.

The effect of the three posterior correction strategies tested (i.e. whether we divide the DNN output likelihoods by some class prior values) is also apparent in Fig. 3. When we performed the Viterbi search by relying on the raw likelihoods (see Fig. 3a), the region $0.4 \leq \lambda \leq 0.6$ proved to be best both for the segment-level and for the frame-level scores. As expected, $\lambda$ values close to 1 proved to be better than those near 0, perhaps with the exception of the ComParE feature set evaluated at the segment level.

When dividing by the original class priors (see Fig. 3b), DNNs trained with $\lambda = 0.1$ or $\lambda = 0.2$ had the highest $F_1$ scores both at the segment level and at the frame level. While for some feature sets there is a drop of accuracy in the region $[0.4, 0.6]$, in general lower $\lambda$ values are better than higher ones. When we transformed the DNN outputs by dividing them by the actual class priors used during training (see Fig. 3c), we can observe a transition between the two previous cases: for $\lambda \geq 0.7$ values, the resulting scores are very close to those obtained via the first strategy; for values $\lambda \leq 0.2$ the scores are close to the second one; while the region in the middle displays a continuous transition between the two. Since $\lambda$ values above 0.6 usually proved suboptimal even in the case of using the original DNN output likelihoods, this strategy did not work well in this region either. For $0.1 \leq \lambda \leq 0.5$, however, it performed consistently better than when we divided by the original class priors, leading to the highest scores overall.

Tables 3 and 4 list the precision, recall and F-measure values obtained on the test set with the optimal $\lambda$ value fine-tuned on the development set, segment and frame-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAFFC.2018.2871450, IEEE Transactions on Affective Computing



TABLE 3
Optimal **segment-level** $F_1$ and the corresponding precision and recall scores on the test set for the two sampling techniques tested

| Feature set | Sampling method | Div. priors | Laughter | | | Filler | | | Combined | | Opt. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Prec. | Rec. | $F_1$ | Prec. | Rec. | $F_1$ | $F_1$ | $p$ | $\lambda$ |
| MFCC | Probabilistic | no | 53.1% | 66.0% | 58.6% | 62.0% | **68.5%** | 65.1% | 62.0% | $= 0.03$ | 0.6 |
| | | yes | 51.7% | 64.1% | 57.0% | 64.3% | 67.3% | 65.7% | 61.5% | $= 0.03$ | 0.2 |
| | | actual | **55.7%** | **67.4%** | **61.0%** | 65.4% | 66.1% | 65.6% | **63.5%** | $< 0.01$ | 0.3 |
| | Downsampling | no | 51.7% | 63.6% | 57.0% | 62.1% | 67.7% | 64.7% | 60.9% | $= 0.03$ | — |
| | Full (baseline) | yes | 49.2% | 57.8% | 53.1% | **65.8%** | 66.0% | **65.9%** | 59.6% | — | — |
| FBANK | Probabilistic | no | **64.5%** | 65.6% | **65.0%** | 67.3% | 65.5% | 66.3% | **65.7%** | $< 0.01$ | 0.5 |
| | | yes | 57.8% | **67.1%** | 61.9% | 67.0% | **67.3%** | **67.0%** | 64.6% | $= 0.03$ | 0.1 |
| | | actual | **64.4%** | 64.1% | 64.2% | 68.9% | 63.8% | 66.2% | 65.3% | $= 0.02$ | 0.2 |
| | Downsampling | no | 51.7% | 63.9% | 57.2% | 64.6% | **67.4%** | 65.9% | 61.7% | — | — |
| | Full (baseline) | yes | 62.9% | 57.3% | 60.0% | **69.2%** | 62.2% | 65.5% | 62.7% | — | — |
| ComParE | Probabilistic | no | **59.7%** | 65.7% | 62.3% | **69.1%** | 60.5% | 64.4% | 63.7% | $= 0.84$ | 0.3 |
| | | yes | 54.9% | **70.0%** | 61.3% | 67.3% | 65.2% | 66.2% | 64.1% | $= 0.69$ | 0.1 |
| | | actual | 58.1% | **69.9%** | **63.1%** | 68.2% | 65.2% | **66.6%** | **65.2%** | $< 0.01$ | 0.2 |
| | Downsampling | no | 54.2% | 66.9% | 59.8% | 67.1% | 66.1% | 66.5% | 63.4% | — | — |
| | Full (baseline) | yes | 58.0% | 62.4% | 60.1% | 66.3% | **67.1%** | **66.7%** | 63.4% | — | — |

TABLE 4
Optimal **frame-level** $F_1$ and the corresponding precision and recall scores on the test set for the two sampling techniques tested

| Feature set | Sampling method | Div. priors | Laughter | | | Filler | | | Combined | | Opt. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Prec. | Rec. | $F_1$ | Prec. | Rec. | $F_1$ | $F_1$ | $p$ | $\lambda$ |
| MFCC | Probabilistic | no | **62.5%** | 56.2% | 59.0% | **58.3%** | 58.1% | 58.1% | 58.6% | $< 0.01$ | 0.6 |
| | | yes | 53.6% | 67.2% | 59.6% | 56.8% | 60.9% | **58.8%** | 59.3% | $< 0.01$ | 0.1 |
| | | actual | 60.5% | 62.2% | **61.3%** | 56.5% | 61.5% | **58.8%** | **60.1%** | $< 0.01$ | 0.2 |
| | Downsampling | no | 46.8% | **72.0%** | 56.7% | 48.1% | **64.2%** | 55.0% | 55.9% | — | — |
| | Full (baseline) | yes | 43.7% | 70.9% | 54.0% | 52.8% | 63.3% | 57.6% | 56.1% | — | — |
| FBANK | Probabilistic | no | **71.7%** | 55.0% | 62.3% | 59.6% | 58.2% | 58.9% | 60.8% | $= 0.03$ | 0.5 |
| | | yes | 58.0% | 71.6% | 64.0% | 58.9% | 61.8% | **60.3%** | **62.3%** | $< 0.01$ | 0.1 |
| | | actual | 63.2% | 66.4% | **64.7%** | **60.1%** | 59.6% | 59.8% | **62.3%** | $< 0.01$ | 0.2 |
| | Downsampling | no | 48.6% | **73.5%** | 58.5% | 47.9% | **64.9%** | 55.1% | 56.8% | — | — |
| | Full (baseline) | yes | 51.9% | 72.3% | 60.4% | 53.8% | 62.4% | 57.8% | 59.2% | — | — |
| ComParE | Probabilistic | no | **66.9%** | 54.9% | 60.2% | **62.5%** | 53.7% | 57.7% | 59.0% | — | 0.3 |
| | | yes | 54.5% | 74.1% | 62.4% | 58.5% | 61.0% | 59.7% | 61.5% | $= 0.02$ | 0.1 |
| | | actual | 60.1% | 68.0% | **63.4%** | 60.2% | 60.4% | **60.2%** | **62.0%** | $< 0.01$ | 0.1 |
| | Downsampling | no | 49.2% | **75.4%** | 59.5% | 52.5% | **64.8%** | 58.0% | 58.9% | — | — |
| | Full (baseline) | yes | 54.6% | 70.4% | 61.4% | 54.3% | 62.6% | 58.1% | 59.8% | — | — |

level scores, respectively. (The best scores are shown in **bold**.) We also indicated the statistical significance ($p$) of the improvements (if any) by using a Mann-Whitney $U$ test [50]. We can see that on the test set, depending on the feature set, the segment-level scores increased by 2.6–3.9% absolute, while the frame-level scores rose by 2.2–4.0% compared to the corresponding baseline values. These gains mean a relative error reduction score of between 5–10% and 5–9% at the segment level and frame level, respectively; the highest scores achieved by utilizing probabilistic sampling (using the FBANK feature set and dividing the output likelihoods by the actual class priors used during training) brought a $7\%$ and a $8\%$ relative error reduction over the baseline. It is interesting to note that most of the improvement came from detecting the laughter events, while the metric values associated with the filler events improved only slightly (or sometimes not at all). Recall (see Table 1) that only $3.4\%$ of the training frames were laughter events, while fillers accounted for $50\%$ more than that ($4.9\%$), hence it is quite understandable that probabilistic sampling helped the detection of the former event types more.

For comparison, we also tested the simple sampling approach of downsampling. We realized downsampling by randomly discarding samples from the "garbage" class to make its frequency match that of filler frames. We got the best results by avoiding the division of the obtained posterior values by the class priors (see tables 3 and 4). Still, the performance of downsampling practically matched that of the baseline, i.e. full database sampling at the segment level, while at the level of frames this strategy even led to somewhat worse $F_1$ values. In our opinion this indicates the downsampling approach is just too simple to improve

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAFFC.2018.2871450, IEEE Transactions on Affective Computing



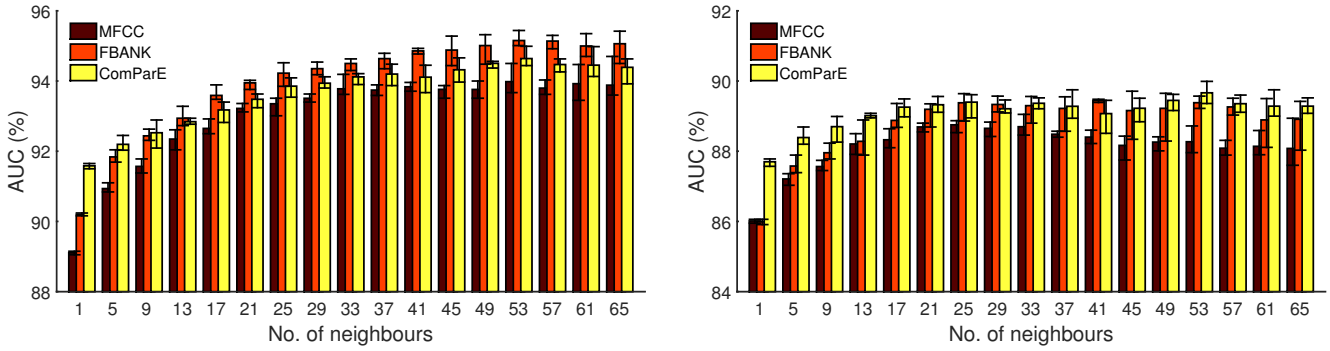


Fig. 4. Averaged AUC values obtained using standard backpropagation DNN training as a function of the number of neighbouring frame vectors used during training, on the development set (left) and on the test set (right).

TABLE 5
The obtained AUC scores using standard backpropagation DNN training for both the development and test sets, and some notable results published in the literature on this dataset

| Approach | Feature Set | No. of frames | Development | | | Test | | |
|---|---|---|---|---|---|---|---|---|
| | | | Lau. | Fil. | Avg. | Lau. | Fil. | Avg. |
| DNN (optimal $F_1$) | MFCC | 21 | 91.6% | 94.8% | 93.2% | 89.7% | 87.6% | 88.7% |
| | FBANK | 29 | 93.3% | 95.4% | 94.4% | 90.6% | 88.1% | 89.3% |
| | ComParE | 33 | 92.7% | 95.5% | 94.1% | 91.0% | 87.7% | 89.4% |
| DNN (optimal AUC) | MFCC | 53 | 92.6% | 95.3% | 94.0% | 90.2% | 86.3% | 88.3% |
| | FBANK | 53 | 94.2% | 96.1% | 95.2% | 91.0% | 87.7% | 89.4% |
| | ComParE | 53 | 93.3% | 96.0% | 94.6% | 91.4% | 88.0% | 89.7% |
| ComParE baseline (Schuller et al. [10]) | | | 86.2% | 89.0% | 87.6% | 82.9% | 83.6% | 83.3% |
| DNN downsampling (Gupta et al. [5]) | | | 90.1% | 90.1% | 90.1% | — | — | — |
| DNN + smoothing + masking (Gupta et al. [5]) | | | 95.1% | 94.7% | 94.9% | 93.3% | 89.7% | 91.5% |
| DNN + DNN (Brueckner and Schuller [34]) | | | 98.1% | 96.5% | 97.3% | 94.9% | 89.9% | 92.4% |
| BLSTM (Brueckner and Schuller [36]) | | | — | — | 97.0% | — | — | 93.0% |
| DNN + GA smoothing (Gosztolya [51]) | | | 97.5% | 96.7% | 97.1% | 96.0% | 90.1% | 93.1% |
| DNN + BLSTM smoothing (Brueckner and Schuller [36]) | | | — | — | 97.2% | — | — | 94.0% |

performance in this task, probably because it reduced the variance of the "garbage" class to a great extent.

Overall, the application of probabilistic sampling gave a significant improvement both at the segment and at the frame levels. (The improvements were found to be significant at the level $p < 0.01$ in all but one case, the only exception being the segment-level $F_1$ values using the FBANK feature set with $p = 0.0159$.) Instead of the extreme values of $\lambda = 0$ and $\lambda = 1$, corresponding to the original and the uniform class distributions, optimal performance was always achieved with intermediate $\lambda$ values. A possible reason is that, since we do not generate further examples of the rarer classes for training, we have to use our samples more frequently. Because of this, we actually use examples belonging to the more frequent classes *less frequently* (actually, not all samples are used for each iteration), which decreases the variance of such classes. By changing the $\lambda$ parameter value, we can establish a trade-off between the two extremes, which seems to work quite well in practice.

## 5 Results Using AUC

Although we primarily measure the accuracy of DNN models trained by calculating segment and frame-level precision, recall and $F_1$ values after using a Hidden Markov Model, as we find this approach more meaningful, next we will also present the frame-level AUC scores obtained for reference.

### 5.1 Baseline Scores

Fig. 4 shows the AUC values obtained on the development and on the test set as a function of the number of neighbouring frame vectors used for all three feature sets tested. We can see that on the development set, the AUC values show an increasing trend even when we use over 50 neighbouring frames, while on the test set the scores stop increasing after about 20 frames, or in the case of the MFCC features they even start to decrease. The reason for this cannot be any form of overfitting, since no meta-parameter setting has yet been applied on the development set; in our opinion, this behaviour suggests there is a mismatch between the development and test sets of this particular database.

Table 5 lists the AUC scores of the DNN models trained via standard backpropagation and full database sampling. We can see that if we chose the number of neighbouring frame vectors based on the optimal $F_1$ value on the development set (i.e. following Fig. 2 and Table 2), we ended up using fewer neighbouring frame vectors (between 21 and 33) than when we chose the size of neighbourhood based on the optimal AUC value on the development set,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAFFC.2018.2871450, IEEE Transactions on Affective Computing



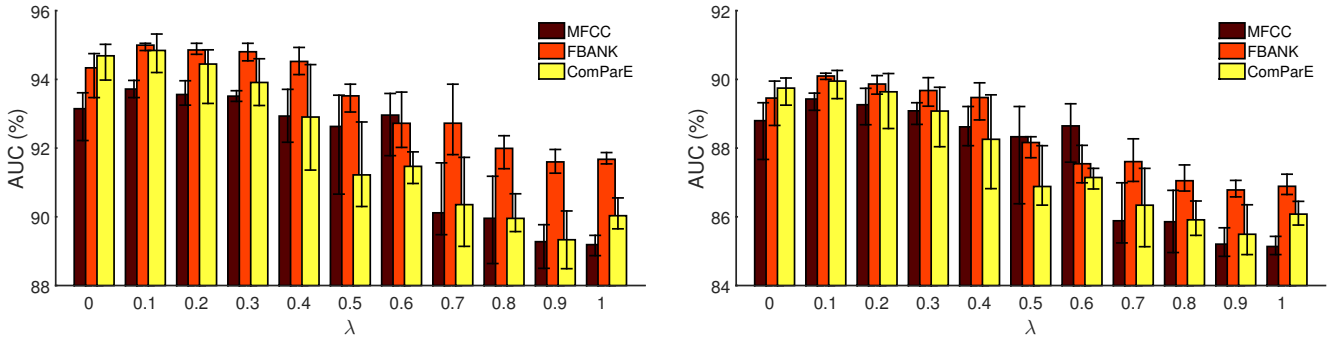


Fig. 5. Averaged AUC scores on the development set (left) and on the test set (right) as a function of $\lambda$.

TABLE 6
Average AUC scores obtained on the development and test sets for all three feature sets tested

| Feature Set | Sampling Approach | Choice of $\lambda$ | Development | | | Test | | | Optimal |
|---|---|---|---|---|---|---|---|---|---|
| | | | Lau. | Fil. | Avg. | Lau. | Fil. | Avg. | $\lambda$ |
| MFCC | Probabilistic | Optimal $F_1$ | 92.5% | 94.6% | **93.6%** | 90.1% | 88.4% | **89.3%** | 0.2 |
| | | Optimal AUC | **93.1%** | 94.4% | **93.7%** | **90.3%** | **88.6%** | **89.4%** | 0.1 |
| | Downsampling | — | 91.2% | 93.5% | 92.4% | 88.6% | 86.9% | 87.7% | — |
| | Full (baseline) | — | 91.6% | **94.8%** | 93.2% | 89.7% | 87.6% | 88.7% | — |
| FBANK | Probabilistic | Optimal $F_1$ | 94.0% | **95.7%** | **94.9%** | 91.2% | **88.6%** | 89.9% | 0.2 |
| | | Optimal AUC | **94.4%** | **95.6%** | **95.0%** | **91.5%** | **88.7%** | **90.1%** | 0.1 |
| | Downsampling | — | 93.0% | 95.0% | 94.0% | 90.3% | 87.6% | 89.0% | — |
| | Full (baseline) | — | 93.3% | 95.4% | 94.4% | 90.6% | 88.1% | 89.3% | — |
| ComParE | Probabilistic | Optimal $F_1$ | **93.9%** | **95.8%** | **94.8%** | **91.8%** | **88.1%** | **90.0%** | 0.1 |
| | | Optimal AUC | **93.9%** | **95.8%** | **94.8%** | **91.8%** | **88.1%** | **90.0%** | 0.1 |
| | Downsampling | — | 93.2% | **95.7%** | 94.5% | 91.2% | **88.0%** | 89.6% | — |
| | Full (baseline) | — | 92.7% | 95.5% | 94.1% | 91.0% | 87.7% | 89.4% | — |

where it was 53 for each feature set. The AUC scores, however, appear to be quite similar: around 92-96% on the development set and around 86-91% on the test set.

Table 5 also shows some notable AUC values published in the literature for this particular dataset. There are two studies which reported AUC scores obtained without any kind of posterior smoothing: both the ComParE baseline, where Schuller et al. used SVM for frame-level vocalization classification [10] and the DNN downsampling approach used by Gupta et al. [5] led to worse scores than our standard DNNs. (Note, however, that these studies did not use any neighbouring feature vectors during classifier training.) It is also clear, however, that posterior smoothing can improve the AUC values by a significant amount; in the other papers listed, the authors employed some kind of smoothing over time on the raw posterior values (time series smoothing and masking [5], [11], smoothing via a second DNN [34], weighted average times series filter optimized via genetic algorithms [51] or smoothing via a recurrent neural network [36]).

### 5.2 Results With Probabilistic Sampling

Figure 5 shows the AUC values achieved on the development and on the test sets using probabilistic sampling, as a function of the $\lambda$ meta-parameter value. We can see that for $\lambda \leq 0.4$, the AUC scores are roughly the same, but there is a visible drop for larger $\lambda$ values. For all three feature sets, we get a slight improvement over the baseline value by using $\lambda = 0.1$ for both the development and the test sets.

Table 6 lists the best AUC scores obtained on both the development and on the test sets for the sampling approaches tested. For comparison, we also showed the AUC scores obtained via DNN training by downsampling and by full database sampling. For probabilistic sampling we listed the cases where AUC was the highest on the development set, and also for those $\lambda$ meta-parameters where we got the highest $F_1$ scores on the frame level (see Table 4). We can see that training our DNN models via downsampling actually made the resulting AUC values worse, compared to the case of full database sampling; the only exception was the case of the ComParE feature set, where we got a slight improvement (0.2% absolute) . Probabilistic sampling, however, improved the scores by a slight amount, increasing the AUC scores by 0.6-0.8% absolute, being equivalent to a 6-8% relative error reduction. There were only slight differences among the two cases of selecting $\lambda$ based on the $F_1$ or AUC scores achieved on the development set. Still, these scores lag behind those listed in Table 5 which employed some posterior smoothing technique. We would like to emphasize, however, that we find measuring the performance of a neural network by utilizing a Hidden Markov Model a more reliable approach for this particular task.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAFFC.2018.2871450, IEEE Transactions on Affective Computing



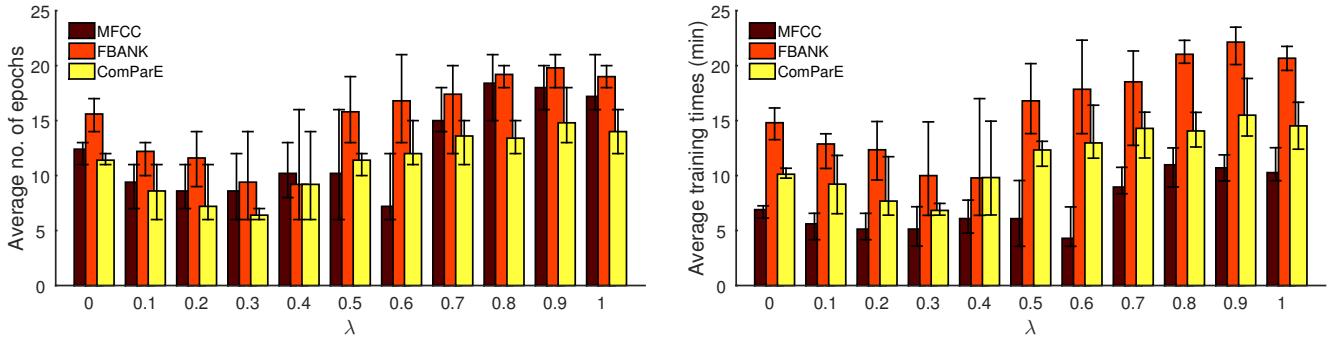


Fig. 6. Average number of training epochs (left) and wall-clock training times (right) as a function of $\lambda$.

TABLE 7
Number of average training epochs and average training times (seconds) as a function of the $\lambda$ values found optimal, and the amount of DNN training speed-up

| Feature set | Sampling method | Div. priors | Segment-level optimality | | | | | Frame-level optimality | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $\lambda$ | Epoch | Impr. | sec. | Impr. | $\lambda$ | Epoch | Impr. | sec. | Impr. |
| MFCC | Probabilistic | no | 0.6 | 7.2 | 42% | 258 | 38% | 0.6 | 7.2 | 42% | 258 | 38% |
| | | yes | 0.2 | 8.6 | 31% | 308 | 26% | 0.1 | 9.4 | 24% | 336 | 19% |
| | | actual | 0.3 | 8.6 | 31% | 308 | 26% | 0.2 | 8.6 | 31% | 308 | 26% |
| | Full (baseline) | yes | — | 12.4 | — | 414 | — | — | 12.4 | — | 414 | — |
| FBANK | Probabilistic | no | 0.5 | 15.8 | -1% | 1008 | -13% | 0.5 | 15.8 | -1% | 1008 | -13% |
| | | yes | 0.1 | 12.2 | 22% | 772 | 13% | 0.1 | 12.2 | 22% | 772 | 13% |
| | | actual | 0.2 | 11.6 | 26% | 741 | 17% | 0.2 | 11.6 | 26% | 741 | 17% |
| | Full (baseline) | yes | — | 15.6 | — | 888 | — | — | 15.6 | — | 888 | — |
| ComParE | Probabilistic | no | 0.3 | 6.4 | 56% | 410 | 33% | 0.3 | 6.4 | 56% | 410 | 33% |
| | | yes | 0.1 | 8.6 | 25% | 553 | 9% | 0.1 | 8.6 | 25% | 553 | 9% |
| | | actual | 0.2 | 7.2 | 37% | 461 | 24% | 0.1 | 8.6 | 25% | 553 | 9% |
| | Full (baseline) | yes | — | 11.4 | — | 607 | — | — | 11.4 | — | 607 | — |

## 6 DNN Training Times

Probabilistic sampling can in theory influence the time spent on DNN model training as well, as using the samples of the rarer classes more frequently could lead to a quicker convergence. Our DNNs are actually trained in a way which exploits this: the accuracy of the networks is measured after each training iteration on a hold-out set, and we cease training when the accuracy score does not improve within a certain number of consecutive iterations (*newbob* learn rate scheduling [52]). When performing probabilistic sampling, we use the same number of training samples for each iteration (*epoch*) as there are present in the whole training dataset. Therefore, we can simply express DNN training times by the number of training iterations, as long as we keep our DNN structure fixed. Another option is to measure the (wall-clock) time spent on the whole training process (including I/O operations, etc.); we will present both values.

The left hand side of Fig. 6 shows the average, minimal and maximal number of training iterations for the different feature sets and $\lambda$ values, while the right hand side shows the measured training wall-clock times. It is clear that for $0.1 \leq \lambda \leq 0.4$, DNN training required fewer iterations than full database sampling to achieve full convergence, which justifies our assumptions. For larger $\lambda$ values, however, the number of training epochs exceeds the baseline value. In our opinion this is caused by a further random factor introduced into DNN training: that of randomly selecting the class of the next training sample. The two figures also reveals that our approach of probabilistic sampling implementation indeed adds only a slight overhead to the training process.

Table 7 shows the average number of training iterations for the different feature sets and optimal $\lambda$ values, and the amount of training speed-up obtained. The 11.4–15.6 epochs required by full database sampling became 6.4–12.2 epochs, reducing the number of training iterations by 22–56% and resulting in a DNN training speed-up of 9–38%. The only exception was the case of the FBANK feature set, where using the DNN output likelihoods directly in the HMM led to an optimal $\lambda$ value of 0.5, but the corresponding $F_1$ scores measured were not so high in this case. For the $\lambda$ values which brought the highest accuracy scores, we needed on average 22–26% fewer training iterations, leading to a 9–17% cut in the actual DNN training times. This means that besides improving the performance, we were also able to train our DNNs faster by using the probabilistic sampling approach, provided that we know the optimal $\lambda$ value in advance. Of course, since for all three feature sets and both evaluation metrics the optimal $\lambda$ lay in the range $[0.1, 0.3]$, we do not necessarily have to try out all possible values. Another way of reducing training times might be to only adapt the baseline DNN model (i.e. train it further) by probabilistic sampling, but this lies out of the scope of our present study.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAFFC.2018.2871450, IEEE Transactions on Affective Computing



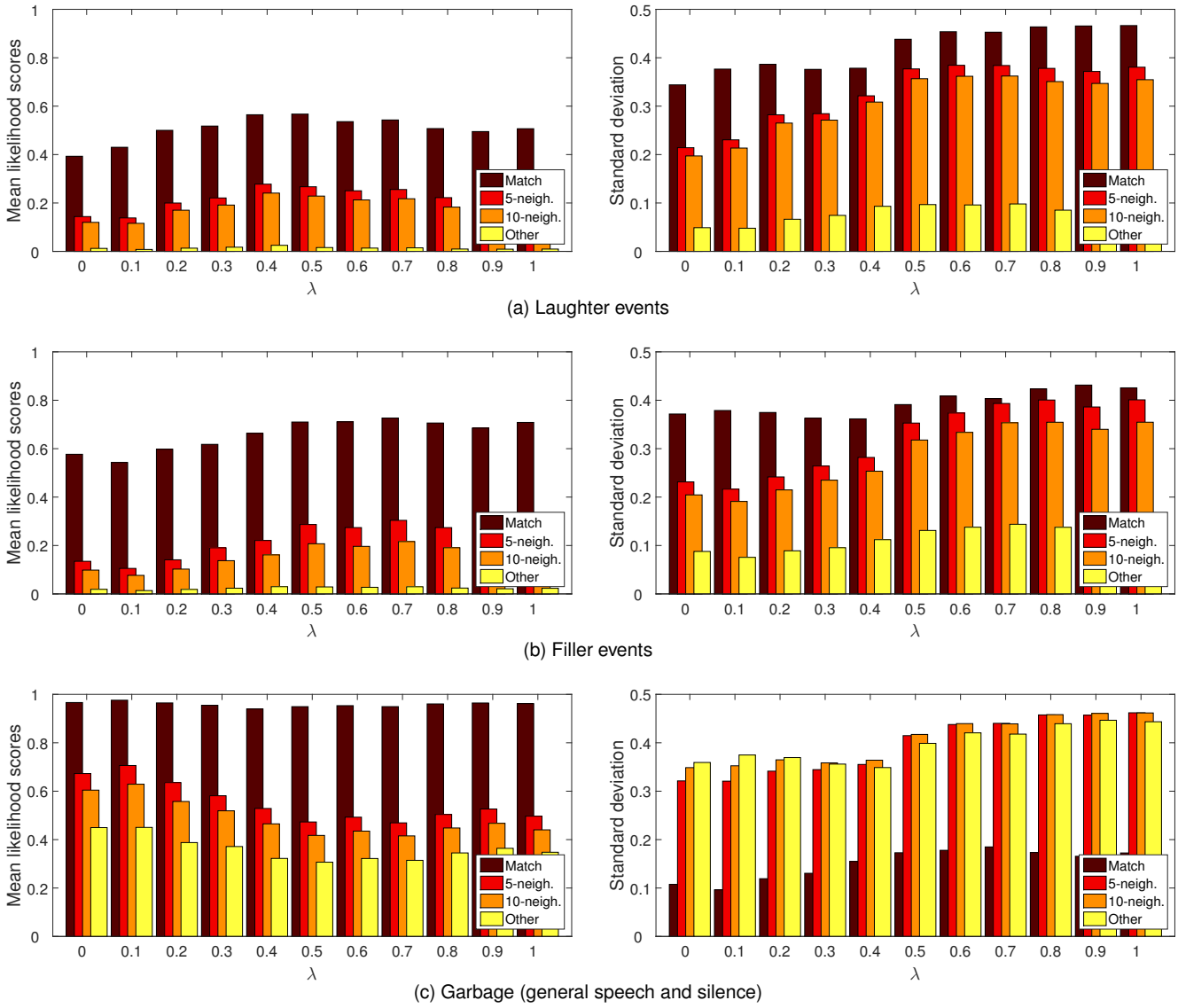


Fig. 7. Mean and standard deviation of the DNN posterior values as a function of $\lambda$ for frames corresponding the given event, frames in the 5 and 10 frame neighbourhood, and for the other frames. The graphs show the values got on the development set using the FBANK feature set.

## 7 Posterior Estimate Statistics

Probabilistic sampling, being an upsampling technique, uses the examples of the rarer classes more frequently in order to increase the corresponding DNN posterior estimates. However, many factors affect the amount of this gain (e.g. learning rate, training epochs, the class distribution, and feature set). Next we will examine the distribution of the posterior scores, concentrating on probabilistic sampling and the effect of the $\lambda$ parameter.

Fig. 7 shows the mean and standard deviation of the posterior estimates for the three classes on the development set; the baseline is again shown as $\lambda = 0$. (The figures only show the statistics for the models trained on the FBANK feature set; models trained on the other two feature sets behaved similarly.) Instead of calculating global values, we took into account the manually annotated frame-level labels during evaluation. Due to reasons given in Section 3.2, we cannot expect a clear-cut boundary at the beginning and the end of an occurrence for the output likelihoods either, but there is probably some kind of a transition present instead. Therefore, besides showing the mean and standard deviation values for the frames associated with the given phenomenon (i.e. laughter, filler or garbage) in the manual annotation, we also calculated them for the 5-frame and 10-frame neighbourhood, as well as for all the remaining frames.

The first thing to notice is that the mean of the posterior estimates for the class "garbage" is pretty high compared to those of the two other classes. However, for the laughter events the mean score hardly ever exceeds $0.5$, even for frames which in fact contain laughter. Although this might be due to the bias present in the class distribution, we just used probabilistic sampling in order to counter this; still, even for $\lambda = 1$ we cannot see much of an improvement. In our opinion this might be due to the different laughter types, and the heterogenous nature of laughter. That is, laughter often contains parts of speech and breath intake [53], [54], which are hard to distinguish from normal speech.

In the five and ten frame neighbourhoods, the posterior

scores are around $0.2$ on average for both of the events we are looking for. This indeed shows a relatively long and soft transition in the likelihood scores, especially compared to the average likelihoods for the other frames. Note that these values lie in the range $[0.4, 0.7]$ for the garbage class, meaning that the posterior score of this class may be dominant even when a laughter or filler event has already begun. Of course, as this class also has the highest a priori probability, applying Bayes' theorem before utilizing these scores in a HMM can mostly counter this effect.

Overall, the DNN output scores associated with the two phenomena we are looking for show a clear trend as a function of the $\lambda$ parameter of probabilistic sampling: in the range $0 \leq \lambda \leq 0.5$ the posterior estimates increase, then there is a drop (for laughter) or they stagnate (for filler events). Examining the standard deviations of the scores, however, we can see that there is a large increase at the point $\lambda = 0.5$ (also for the frames in the neighbourhood of the annotated occurrences), and they keep on increasing for $\lambda > 0.5$. This, in our opinion, explains why lower $\lambda$ values turned out to be optimal: while the mean posterior scores within and near an occurrence are higher than what was produced by the baseline models, their standard deviation is not higher; or, in the case of filler events, only slightly so. In contrast, for larger $\lambda$ parameter values the likelihood scores are generally higher, but they are also less reliable, which is reflected by both the standard deviation scores shown in Fig. 7 and by the $F_1$ scores shown in Fig. 3.

## 8 Conclusions

In this study, we sought to balance the distribution of DNN training data in social signal detection in speech. We chose the technique called probabilistic sampling, which makes adjusting the class distribution of the samples between the original distribution and a uniform distribution possible with the help of a parameter. We also proposed an efficient way for implementing this algorithm. Our experiments showed that this approach is more effective than DNN training using either traditional full database sampling or downsampling: following the probabilistic sampling strategy we were able to achieve improvements between 5 and 10% both in the $F_1$ scores and in the AUC values on the test set of a public database containing spontaneous English mobile phone conversations. We also showed that this sampling technique can aid DNN training regardless of the feature set used, as we got improvements for three different standard frame-level feature sets. We also experienced a drop in DNN training times in the range 24–67%, provided that we know the optimal parameter value of probabilistic sampling in advance. Probabilistic sampling allows us to further speed up training DNN acoustic models by just adapting a pretrained DNN via probabilistic sampling; this, however, is clearly the subject of future research.

## Acknowledgments

Gábor Gosztolya was funded by the National Research, Development and Innovation Office of Hungary via contract ID-124413 'Enhancement of deep learning based semantic representations with acoustic-prosodic features for automatic spoken document summarization and retrieval'. Tamás Grósz was supported by the ÚNKP-17-3 new national excellence programme of the ministry of human capacities of the Hungarian government. László Tóth was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

## References

[1] L. S. Kennedy and D. P. W. Ellis, "Laughter detection in meetings," in *Proceedings of the NIST Meeting Recognition Workshop at ICASSP*, Montreal, Canada, 2004, pp. 118–121.

[2] Y.-X. Li and Q.-H. He, "Detecting laughter in spontaneous speech by constructing laughter bouts," *International Journal of Speech Technology*, vol. 14, no. 3, pp. 211–225, 2011.

[3] T. Neuberger, A. Beke, and M. Gósy, "Acoustic analysis and automatic detection of laughter in Hungarian spontaneous speech," in *Proceedings of ISSP*, 2014, pp. 281–284.

[4] H. Salamin, A. Polychroniou, and A. Vinciarelli, "Automatic detection of laughter and fillers in spontaneous mobile phone conversations," in *Proceedings of SMC*, 2013, pp. 4282–4287.

[5] R. Gupta, K. Audhkhasi, S. Lee, and S. S. Narayanan, "Speech paralinguistic event detection using probabilistic time-series smoothing and masking," in *Proceedings of Interspeech*, 2013, pp. 173–177.

[6] G. Gosztolya, R. Busa-Fekete, and L. Tóth, "Detecting autism, emotions and social signals using AdaBoost," in *Proceedings of Interspeech*, 2013, pp. 220–224.

[7] D. Prylipko, O. Egorow, I. Siegert, and A. Wendemuth, "Application of image processing methods to filled pauses detection from spontaneous speech," in *Proceedings of Interspeech*, 2014, pp. 1816–1820.

[8] I. Siegert, M. Haase, D. Prylipko, and A. Wendemuth, "Discourse particles and user characteristics in naturalistic human-computer interaction," in *Proceedings of HCI*, 2014, pp. 492–501.

[9] I. Siegert, J. Krüger, M. Haase, A. F. Lotz, S. Günther, J. Frommer, D. Rösner, and A. Wendemuth, "Discourse particles in human-human and human-computer interaction – Analysis and evaluation," in *Proceedings of HCI*, 2016, pp. 105–117.

[10] B. Schuller, S. Steidl, A. Batliner, A. Vinciarelli, K. Scherer, F. Ringeval, M. Chetouani, F. Weninger, F. Eyben, E. Marchi, H. Salamin, A. Polychroniou, F. Valente, and S. Kim, "The Interspeech 2013 Computational Paralinguistics Challenge: Social signals, Conflict, Emotion, Autism," in *Proceedings of Interspeech*, 2013.

[11] R. Gupta, K. Audhkhasi, S. Lee, and S. S. Narayanan, "Detecting paralinguistic events in audio stream using context in features and probabilistic decisions," *Computer, Speech and Language*, vol. 36, no. 1, pp. 72–92, 2016.

[12] A. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[13] J. Holmes and M. Marra, "Having a laugh at work: How humour contributes to workplace culture," *Journal of Pragmatics*, vol. 34, no. 12, pp. 1683–1710, 2002.

[14] N. V. Chawla, *Data Mining and Knowledge Discovery Handbook*. Boston, MA: Springer, 2005, ch. Data Mining for Imbalanced Datasets: An Overview, pp. 853–867.

[15] S. J. Young, J. J. Odell, and P. C. Woodland, "Tree-based state tying for high accuracy acoustic modelling," in *Proceedings of HLT*, 1994, pp. 307–312.

[16] W. Wang, H. Tang, and K. Livescu, "Triphone state-tying via deep canonical correlation analysis," in *Proceedings of Interspeech*, San Francisco, CA, USA, Sep 2016, pp. 3444–3448.

[17] G. Gosztolya, T. Grósz, L. Tóth, and D. Imseng, "Building context-dependent DNN acousitc models using Kullback-Leibler divergence-based state tying," in *Proceedings of ICASSP*, Brisbane, Australia, 2015, pp. 4570–4574.

[18] C. Peláez-Moreno, Q. Zhu, B. Y. Chen, and N. Morgan, "Automatic data selection for MLP-based feature extraction for ASR," in *Proceedings of Interspeech*, Lisboa, Portugal, 2005, pp. 229–232.

[19] T. Ha and H. Bunke, "Off-line, handwritten numeral recognition by perturbation method," *Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp. 535–539, 1997.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with Deep Convolutional Neural Networks," in *Proceedings of NIPS*, 2012, pp. 1097–1105.

[21] R. Mash, B. Borghetti, and J. Pecarina, "Improved aircraft recognition for aerial refueling through data augmentation in Convolutional Neural Networks," in *Proceedings of IVSC*, Las Vegas, NV, USA, 2016, pp. 113–122.
[22] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proceedings of Interspeech*, Dresden, Germany, Sep 2015, pp. 3586–3589.
[23] W. Hartmann, T. Ng, R. Hsiao, S. Tsakalidis, and R. Schwartz, "Two-stage data augmentation for low-resourced speech recognition," in *Proceedings of Interspeech*, San Francisco, CA, USA, Sep 2016, pp. 2378–2382.
[24] A. Ragni, K. M. Knill, S. P. Rath, and M. J. F. Gales, "Data augmentation for low resource languages," in *Proceedings of Interspeech*, Singapore, Singapore, 2014, pp. 810–814.
[25] X. Cui, V. Goel, and B. Kingsbury, "Data augmentation for Deep Neural Network acoustic modeling," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 23, no. 9, pp. 1469–1477, 2015.
[26] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 2002, no. 16, pp. 321–357, 2002.
[27] S. Lawrence, I. Burns, A. Back, A. Tsoi, and C. Giles, "Chapter 14: Neural network classification and prior class probabilities," in *Neural Networks: Tricks of the Trade*. Springer, 1998, pp. 299–313.
[28] L. Tóth and A. Kocsor, "Training HMM/ANN hybrid speech recognizers by probabilistic sampling," in *Proceedings of ICANN*, 2005, pp. 597–603.
[29] A. I. García-Moral, R. Solera-Urena, C. Peláez-Moreno, and F. D. de María, "Data balancing for efficient training of hybrid ANN/HMM Automatic Speech Recognition systems," *IEEE Transactions on Audi, Speech and Language Processing*, vol. 19, no. 3, pp. 468–481, 2011.
[30] T. Grósz, G. Gosztolya, and L. Tóth, "Training context-dependent DNN acoustic models using probabilistic sampling," in *Proceedings of Interspeech*, Stockholm, Sweden, Aug 2017, pp. 1621–1625.
[31] A. Newell and J. C. Shaw, "Programming the logic theory machine," in *Proceedings of WJCC*, Los Angeles, CA, USA, 1957, pp. 1–51.
[32] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. Wiley & Sons, New York, 1973.
[33] H. Kaya, A. Ercetin, A. Salah, and S. Gürgen, "Random forests for laughter detection," in *Proceedings of WASSS*, 2013.
[34] R. Brueckner and B. Schuller, "Hierarchical neural networks and enhanced class posteriors for social signal classification," in *Proceedings of ASRU*, 2013, pp. 362–367.
[35] G. Gosztolya, "On evaluation metrics for social signal detection," in *Proceedings of Interspeech*, Dresden, Germany, Sep 2015, pp. 2504–2508.
[36] R. Brueckner and B. Schuller, "Social signal classification using deep BLSTM recurrent neural networks," in *Proceedings of ICASSP*, 2014, pp. 4856–4860.
[37] L. Kaushik, A. Sangwan, and J. H. L. Hansen, "Laughter and filler detection in naturalistic audio," in *Proceedings of Interspeech*, 2015, pp. 2509–2513.
[38] C. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.
[39] D. Castán, D. Tavarez, P. Lopez-Otero, J. Franco-Pedroso, H. Delgado, E. Navas, L. Docio-Fernández, D. Ramos, J. Serrano, A. Ortega, and E. Lleida, "Albayzín-2014 evaluation: Audio segmentation and classification in broadcast news domains," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2015, no. 33, pp. 1–10, 2015.
[40] A. Beke, "Automatic speaker diarization," Ph.D. dissertation, Eötvös Lóránt University, Budapest, Hungary, 2014.
[41] F. B. Pokorny, R. Peharz, W. Roth, M. Zöhrer, F. Pernkopf, P. B. Marschik, and B. Schuller, "Manual versus automated: The challenging routine of infant vocalisation segmentation in home videos to study neuro(mal)development," in *Proceedings of Interspeech*, San Francisco, CA, USA, Sep 2016, pp. 2997–3001.
[42] *NIST Spoken Term Detection 2006 Evaluation Plan*, http://www.nist.gov/speech/tests/std/docs/std06-evalplan-v10.pdf, 2006.
[43] L. Tóth, "Phone recognition with hierarchical Convolutional Deep Maxout Networks," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2015, no. 25, pp. 1–13, 2015.
[44] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.
[45] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.
[46] R. Cai, L. Lu, H.-J. Zhang, and L.-H. Cai, "High-light sound effects detection in audio stream," in *Proceedings of ICME*, 2003, pp. 37–40.
[47] G. Gosztolya, A. Beke, T. Neuberger, and L. Tóth, "Laughter classification using Deep Rectifier Neural Networks with a minimal feature subset," *Archives of Acoustics*, vol. 41, no. 4, pp. 669–682, 2016.
[48] S. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book*. Cambridge, UK: Cambridge University Engineering Department, 2006.
[49] F. Eyben, M. Wöllmer, and B. Schuller, "Opensmile: The Munich versatile and fast open-source audio feature extractor," in *Proceedings of ACM Multimedia*, 2010, pp. 1459–1462.
[50] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50–60, 1947.
[51] G. Gosztolya, "Optimized time series filters for detecting laughter and filler events," in *Proceedings of Interspeech*, Stockholm, Sweden, Aug 2017, pp. 2376–2380.
[52] C. Zhang and P. C. Woodland, "A general Artificial Neural Network extension for HTK," in *Proceedings of Interspeech*, Dresden, Germany, Sep 2015, pp. 3581–3585.
[53] J. Trouvain, "Phonetic aspects of "speech-laughs"," in *Proceedings of Orage*, Aix-en-Provence, France, 2001, pp. 634–639.
[54] T. Neuberger and A. Beke, "Automatic laughter detection in Hungarian spontaneous speech using GMM/ANN hybrid method," in *Proceedings of SJUSK Conference on Contemporary Speech Habits*, 2013, pp. 1–13.

**Gábor Gosztolya** received his MSc and PhD degrees from the University of Szeged, Szeged, Hungary in 2001 and 2010, respectively. He is working at the University of Szeged, and at the Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and the University of Szeged. His research interests include speech recognition, computational paralinguistics, various speech technologies, and applied machine learning.

**Tamás Grósz** received his BSc and MSc degrees from the University of Szeged, Szeged, Hungary in 2011 and 2013, respectively. He is currently a PhD student at the University of Szeged. His research interests are speech recognition, deep learning technologies and image processing.

**László Tóth** is a senior researcher at the Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and the University of Szeged. He received his M.Sc. and Ph.D. degrees from the University of Szeged. He joined the Research Group on Artificial Intelligence in 1995, where his main research interests are speech recognition, machine learning and signal processing. Currently, he is focusing on the application of deep neural networks (DNNs) to speech recognition, including the most recent DNN technologies such as convolutional and recurrent neural networks.