Human Computer Interaction Improvement by Interjection Recognition: A new speech processing task and dataset

Yaniv Goren, Alexander Apartsin, and Yehudit Aperstein

Intelligent Systems Department, Afeka Tel Aviv Academic College of Engineering

[Word count]

Human Computer Interaction Improvement by Interjection Recognition: A new speech processing task and dataset

**Abstract.** One of the main challenges ofsuccessful human-computer interaction is enabling a natural and spontaneous dialogue between human and machine such as that experienced in human-to-human dialogue.Although the use of interjections (e.g., “mmm”, “hmm”) convey important information in colloquial speech, they are usually considered to be “non-words” by Automatic Speech Recognition (ASR) engines. Recognizing and supporting interjection in speech-based user interfaces (e.g., voice control) could result in much more natural human-machine interactions. Moreover, interjection recognition can be utilized for Speech Emotion Recognition (SER) in call center services, including those for emergencies. In this work, we present a dataset of interjection audio records collected for the task of interjection recognition. The collected dataset is used to train and evaluate two baseline neural network models on inter-speaker and intra-speaker interjection classification. To improve performance, the collected dataset was extended using a mix of several augmentation techniques (e.g., tempo and pitch transformation). We show that the training based on the augmented dataset results in a significant improvement in the classification accuracy and reduces the need for a large number of records to train the models.

Introduction

People tend to choose an efficient form of verbal communication by leveraging common understanding of context between two parties. Interjections are one of the major parts of speech frequently used to convey meaning in a specific context. Many smart systems include a speech recognition system that enables natural dialogue between man and machine (Hoy, 2018). Among well-known examples are virtual personal assistants, including Apple's Siri artificial intelligence system (Berdasco, et. al., 2019), or Amazon’s Alexa (Lopatovska et. al, 2019). A Siri or Alexa user can, with natural speech queries, obtain information and perform various actions in several domains (e.g., checking weather or stock prices, ordering pizza, etc.).

In recent years, a communications revolution has expanded from basic people-to-people communication to include people to-machine communications. In order to succeed, this revolution demands a high-quality interface to support successful Human Computer Interactions (HCI) (Clark et. al, 2019). However, to create successful speech-enabled applications, those systems must overcome the limitations of both current speech technologies and human cognitive processing. The challenge is not only to advance technology performance, but also to understand how to integrate these technologies into viable, easy-to-use spoken language systems.

One of the limitations in HCI is that, although spontaneous conversation is optimized for human-human interactions, it differs from the types of speech for which human language technology is often developed. Adding interjection recognition capabilities to voice assistants will improve human computer interactions and increase usage, with spontaneous conversation in human-machine interfaces. The improvement can be expressed by interpreting the meaning of hard-to-understand speech, such as a heavy accent, or a sentence in which not all words are clear (Gouda et. al., 2018). By understanding prominent keywords, we can understand the whole sentence. For example, when a human says “Oy, …” in a conversation, he wants to express an unexpected situation, something unfortunate, or perhaps fright.

Furthermore, expressing a spontaneous feeling is one of the key features of most interjections and it can be utilized for Speech Emotion Recognition (SER) (Khalil et.al., 2019). While humans can efficiently recognize the emotional aspects of speech, this ability in machines is still an ongoing subject of research. Enabling machines to understand emotions can provide efficient methods of detecting emotions through different call center services, emergency call centers, and many other human-machine communication users. The capability of machines to detect emotions and act accordingly is a critical factor of making machines appear and act in a human-like manner.

In another sense, voice interjections can be considered as a sort of “voice touch” signal. Similar to immensely successful haptic touch interfaces, the interjection-based voice touch interfaces might enable effortless human machine interactions with voice assistants and other voice-enabled devices. For some applications, tasks can be accomplished successfully by identifying an interjection and then mapping it to an appropriate action or response. The interjection can be considered as a shortcut to a repeated action that requires more effort from the user and releases him from stating the request to the machine in a detailed way. For example, consider a user that regularly asks his personal assistant, “What time is the next train to King's Cross from my closest station?”, or “How long will it take me to drive home?” Or another user that, every time he enters his car after work, sets his GPS for the drive home and immediately calls home to his wife. By giving the user the ability to pick an interjection phrase from a set and map it to a desired action, such interactions could be facilitated by customizing and adapting the system to the needs and preferences of individual users.

It is important to emphasize that our goal is not to detect a specific interjection phrase within speech, although a considerable amountof interjections have a semantic meaning by themselves without the context of a conversation. For example, “A-ha” is a consent regardless of the context, and “wow” express a strong feeling or astonishment. Moreover, many phrases are considered to be interjections and we are not trying to build a complete system that can recognize all interjections. The purpose of this document is to present a system that will serve as a benchmark interjection identifier system and a solid foundation for adding new interjections quickly and simply. Our motivation is to enrich user interface technologies that enable system designers to create habitable human machine interfaces and dialogues that maintain natural interactions with the machine.

The collection and the preparation of the training data is a major challenge for this project. The available speech datasets are focused on word level (Warden, 2018), phoneme level (Proutskova et. al., 2012), or event level (Imoto, 2018) tasks, while interjections lie somewhere in-between. Therefore, we collected our own unique baseline dataset by recording a relatively large set of interjections and negative examples from some speakers.

In this work, we propose a neural network model for interjection recognition and classification. We collected datasets and used them to train, evaluate and test the models. An interjection recognizer accepts a waveform feature and returns K+1 labels, where K is the number of supported interjection classes. The additional class label is reserved for non-interjection audio input. It is known that deep learning requires a large amount of data to train an accurate model. To increase the amount of training data and reduce overﬁtting, we enriched the dataset by augmenting the original data through the application of various artificial distortions (Zhou et. al., 2017). The data augmentation includes the addition of background noise (Richey et. al, 2018) and pitch and tempo modifications (Kulkarni & Naik, 2018).

In this paper, we present the results of several different interjection recognition baseline experiments, where relative improvement was obtained by using the proposed data augmentation methods over a state-of-the-art Feedforward Neural Network (FNN) and Recurrent Neural Network (RNN) classification models. Our data augmentation methods have been implemented in the training data. By training the network on the additional deformed data, the network becomes invariant to these deformations and generalizes better to unseen data.

The rest of the paper is organized as follows: The next section reviews previous work with an emphasis on speech recognition, augmentation methods in speech recognition, and interjections-related research. This is followed by an explanation of our contribution. Next, we describe the dataset creation process and description of our data augmentation tool, followed by the interjection recognition models and the experimental setup. We then discuss the results and provide a summary of the proposed method with some possible research directions.

Previous Work

Studies regarding the processing and understanding of speech and voice signals occur in several contexts. One of the most common tasks associated with processing speech and voice signals is Automatic Speech Recognition (ASR) (Zerari et. al., 2016).

ASR is the task of translating audio signals into text. The main challenge is to overcome the non-stationarity of the speech signal and the large variations in its spatiotemporal representation. As illustrated in Figure 1, the typical ASR system usually includes several steps. The pre-processing step, where an analog speech signal is transformed into a digital signal, includes speech/non-speech segmentation and filtering. The feature extraction step deals with the transformation of the incoming digital waveform into a vector representation of desirable speech features that emphasize linguistic information. Usually, a speech signal is broken into short (usually around 20-30 ms) segments (frames), overlapped every 10 ms, during which the signal is assumed to be stationary (Majeed et. al, 2015). During the last few decades, several techniques have been developed for feature extraction from speech signals. These approaches include Mel-Frequency Cepstral Coefficients (MFCCs), Perceptual Linear prediction (PLP), Relative Spectral (RASTA), and Linear Predictive Coding (LPC) (Gadekar et. al., 2019). However, MFCCs are probably used most commonly (Majeed et. al, 2015).

MFCCs extract spectral features by defining an analysis window (around 25 ms) and divide the speech signal into different time frames by shifting the window with a 10 ms shifting stride. Then Fast Fourier Transformation (FFT) is calculated for each frame to obtain the frequency features, and the logarithmic Mel-Scaled filter bank is applied to its power spectrum estimate. MFCCs calculate the Discrete Cosine Transformation (DCT) of the log energies in the corresponding frequency bands to obtain an m-dimensional coefficients vector. The measured power spectrum envelope in each frame correlates to the shape of the vocal tract, providing an appropriate representation of the sound or phoneme being produced. This procedure results in feature vectors that can be arranged in a [n×m] matrix, where m is the number of coefficients and n is the number of frames.

At the heart of an ASR system is the decoder. During this phase, feature vectors are decoded into linguistic units that make up speech. The decoding relies on acoustic and language models (Ghai & Singh, 2012) to recover the most probable utterance by modeling the conditional probability of the nth word, using the (n-1) earlier words. Linguistic and pronunciation dictionaries are often used to improve the decoding performance. An acoustic model (Ghai & Singh, 2012) is a fundamental part of the ASR system, where the connection between the acoustic information (e.g., feature vectors) and phonetics is established. The acoustic models are frequently implemented using various methods that include the Hidden Markov Model (HMM) (Gruhn et. al., 2011) and support vector machines (SVM) (Pradhan, 2012). HMM is the most commonly used acoustic model for speech recognition in many practical applications.

**Figure 1 around here**

ASR is a broad topic that includes sub-topics related to interjection recognition. One of those sub-topics is Keyword Spotting (KWS) (Chandra & Senthildevi, 2015). The need to provide users with a fully hands-free experience for situations like driving resulted in the development of a system that listens continuously for specific keywords to initiate voice input. Keyword Spotting aims at detecting predefined keywords in an audio stream. A commonly used technique for keyword spotting is the Keyword/Filler Hidden Markov Model (HMM). One disadvantage with this technique is that it can be computationally expensive, and the model is trained separately for each keyword (Chen et. al., 2014).

Keyword spotting is sometimes performed using a pattern matching approach where the input is compared with a few pre-recorded commands. Since the same word might be articulated with a different speed, the Dynamic Time Warping (DTW) algorithm is used to align two sequences in an optimal way (Yadav & Alam, 2018). Recent neural network models show a significant improvement over the HMM approach. The recurrent neural networks (RNN) used in the Deep KWS model (Chen et. al., 2014) shows good performance while keeping a reduced runtime computation and smaller memory footprint. A Convolutional Neural Network (CNN) architecture described by Sainath and Parada (2015) shows improvements over FNN in a variety of small and large vocabulary tasks. The described CNN architecture generalizes more easily to different speaking styles compared to a fully connected FNN architecture.

Another sub-topic related to interjection recognition is Audio Event Detection (AED) (Imoto, 2017), which is considered a common task of processing speech and voice signals. An audio event is a specific type of sound, such as footsteps, running water, exhaust fan noise, screams, ocean waves breaking, or music. Many sound clips contain multiple acoustic events that overlap on the time axis. AED is a task in which a relatively long (several seconds to tens of seconds) sound clip including multiple acoustic events serves as input, and the output is acoustic event labels and their time stamps (start and end times). The process of AED is created by extracting acoustic features using MFCC in general, and then constructing classiﬁcation models using Gaussian mixture models (GMM) (Eronon et. al., 2005), HMM (Chun et. al., 2013), SVM (Geiger et. al., 2013) or more recently models such as CNN (Valenti et. al., 2016) or RNN (Bae et. al., 2016). Because acoustic events can often have a temporal overlap, the most diﬃcult problem in AED is how to detect active durations of acoustic events. For instance, Lee et al. (2017) proposed acoustic event detection based on a convolutional neural network, which calculates a posterior for the existence of acoustic events time frame by time frame.

Generally, deep learning requires a large amount of labeled training data to enable accurate speech recognition. To the best of our knowledge, such a large dataset does not exist for interjections. Therefore, data augmentation is proposed, where the speech data are artificially augmented by applying different types of distortions in a way that does not change the label. Ragni et. al. (2014) proposed data augmentation for low resource speech recognition tasks. The performance of an FNN speech recognition model depends on how well the training data matches the testing data. This can be done by increasing the generalization of a model beyond the data provided to it, and by attempting to create similarity between the training data and the representative characteristics that are seen in real data, such as voice variability of different speakers, or different background noises.

There are many options for data augmentation. Some of them are applied in the feature level of a neural network, and some are applied directly in the raw audio level. Vocal tract length perturbation (VTLP) (Jaitly & Hinton, 2013) is a popular method for doing feature level data augmentation in speech, and has shown gains on the TIMIT (a collection of phonemically transcribed [American English](https://en.wikipedia.org/wiki/American_English" \o "American English) speech) phoneme recognition task. SpecAugment (Park et. Al., 2019) is another feature level data augmentation method for speech recognition that operates on the log Mel spectrogram of the input audio. In the raw audio level, intuitive and practical transformations such as Dynamic Range Compression (DRC), pitch-shifting, time stretching, and background noise combination apply audio effects to the original training audio files (Salamon & Bello, 2017).

Despite the prevalence of interjections in human speech patterns, interjections in the literature are mentioned mainly in the context of emotion where researchers seek to understand the nature of interjections and interjectional meaning (Goddard, 2014). There is a difference between formal speech and conversational speech. Conversational speech is more spontaneous and efficient and does not require special training. Elizabeth Shriberg (2005) describes four fundamental properties of spontaneous speech that present problems for spoken language applications, such as lack of punctuation or the inability to “hear” a speaker’s emotion through speech. As stated before, SER is still an ongoing subject of research and the main traditional techniques for SER are based on feature extraction and selection to identify various emotions (Khalil et. al., 2019).

No prior work, to our knowledge, has explored inserting interjections in speech recognition systems. Cohn et. al. (2019) described an experiment that systematically manipulated the Amazon Alexa TTS by adding some emotional-cognitive expressions, like interjections, while also examining the influence of interjection duration and pitch levels. The conclusion was that those manipulations improved user’s ratings of their conversation across thousands of real user interactions. Likewise, an emotional speech recognition system that classified “fear emotion” for systems like emergency call centers is described by Yoon et. al. (2019). This system used Support Vector Machines (SVM) with an interjections feature, that classified a type of incredulity observed in spontaneous speech when a speaker gets hyperemotional. It is important to clarify that this system cannot recognize different interjections but rather knows how to classify calls into two classes: calls that include fear and calls that do not include fear.

Our Contribution

We represent a different way of looking at the interjection classification task. We implement the classification using a fully connected feedforward neural network (FNN) and Long Short-Term Memory (LSTM) . It can be challenging to identify a sound that is a short fragment, has no context, and could be pronounced in various ways by the same speaker, and differently by other speakers.

To the best of our knowledge, there is no dataset of interjections that can be used for the research. In this work we create a new dataset of interjection phrases using different speakers. In addition, we propose the use of audio data augmentation to overcome the disadvantage of data scarcity and explore the influence of different augmentation sets on the performance of the proposed architecture. By training the network on the additional augmentation data, we make the network invariant to these deformations and more generalizable to unseen data.

Method

*Dataset*

The lack of an interjection dataset required us to create one. The dataset includes “clean” unsynthesized audio samples of the five labels by five different speakers (two females and three males) without background noises. The dataset was then expanded using the augmentation process that artificially modified pitch, tempo, and background noise, in addition to existing recordings.

### *New Dataset Creation*

### One of the prerequisites of deep learning is a high-quality collection of data that can be used for training. For this project, annotated audio datasets are needed, consisting of short sound clips, and accompanying labels that tell us the subject of the recording. This would identify each sound clip as belonging to one of a finite set of categories and enable the problem to be tackled as a supervised learning task. We choose four examples of interjections (Table 1) for this study. They were chosen since they are independent of language and have a semantic meaning by themselves and do not require a conversation for context. The fifth class of non-interjection words consists of words selected by reading from several books in a completely random order, and vary by speaker.

**Table 1 around here**



The sound clips were recorded using Auditok (https://github.com/ramya1782/auditok), which is a VAD (Voice Activity Detection) tool (Sahidullah & Saha, 2012) that enables recording and saving each sound clip as a separate wav file, with a 16000 Hz sampled rate. At the end of that process, five folders (four interjections and one negative example) were created with five speakers each. Table 2 lists the profile and the number of audio samples recorded per speaker. The column “Number of Audio Samples” refers to a specific word, e.g. Speaker A has 850 audio samples per word and a total number of 4,250 audio samples.

**Table 2 around here**



### *New Dataset Creation*

### To extract the useful features from the audio file, we used Librosa library (McFee et. al. 2015). This library is a python package for music and audio analysis that provides several methods to retrieve information from sound clips.

### Feature extraction for FNN model

### A minimum and maximum recording length was determined for dataset samples. The minimum length is 0.45 seconds, so as to not allow short and unrelated recordings, such as background noises that were mistakenly recorded separately. The maximum length is 1.55 seconds, in case quiet was not detected by Auditok between recordings. Recordings with lengths not in the range between the minimum and maximum length were removed.

The methods from Librosa used to extract various features and the number of features extracted for each method are:

* ***MFCC (Librosa.feature.mfcc)*** - 40 features.
* ***Melspectrogram (librosa.feature.melspectrogram)*** - 128 features.
* ***Chorma-stft (librosa.feature.chroma\_stft)*** - 12 features.
* ***Spectral contrast (librosa.feature.spectral\_contrast)*** - 7 features.
* ***Tonnetz (librosa.feature.tonnetz)*** - 6 features.

The result of that process is a matrix with a row for each sample audio file, and a column for each mean feature value (193 columns).

### Feature extraction for LSTM

#### All recordings remained at their original length, filtered like the recordings in FNN by a minimum length of 0.45 seconds and maximum recording length of 1.55 seconds. MFCC from Librosa was used with a default sample rate of 22 kHz and analysis window of 10 kHz. Each window was divided into 21 frames and 40 coefficients were extracted from each frame. The result of that process is a three-dimensional matrix [w, 21 (number of frames), 40 (number of coefficients)] where w is the number of windows.

*Data Augmentation System (DAS)*

To extend the collected dataset, a data augmentation system (DAS) that generates synthetic samples was developed. We used Pysox (Bittner et. al., 2016), a Python library that provides a simple interface between Python and Sox, a popular command line tool for sound processing that can apply various effects to audio files. We tried four different sound effects on our original unsynthesized (“clean data”) audio samples. Each effect was applied directly to the sound file prior to converting it into the input vector representation for the neural network. The data augmentation effects are described below:

* ***Tempo***: Changes the audio playback speed but not its pitch. This function is a parameter ‘factor’ (factor > 1 speeds up the audio signal, factor < 1 slows down the audio signal). Duration of the sound file changes.
* ***Pitch***: Changes the audio pitch (but not tempo). The sensation of a frequency is commonly referred to as the pitch of a sound. A high-pitch sound corresponds to a high-frequency sound wave and a low-pitch sound corresponds to a low-frequency sound wave. One octave (the [interval](https://en.wikipedia.org/wiki/Interval_(music)" \o "Interval (music)) between one musical [pitch](https://en.wikipedia.org/wiki/Pitch_(music)" \o "Pitch (music)) and another) is divided into 12 semitones (tones) of 100 cents each. Typically, cents are used to express small intervals. 1200 cents equal one octave. This function gets the parameter ‘n-semitones’ (the positive or negative number of semitones to shift).
* ***Background noise***: Mixes the original “clean” audio with another recording containing background sounds from several acoustic scenes. Each sample is mixed with nine acoustic scenes: baby gibberish, ambulance, crowd laughing, football crowd, mall, passing bus, rain, street traffic, and TV. Each mix Smix was generated using Equation 2

Smix = (Worig × Sorig) + (Wbackground × Sbackground) (1)

where Sorig is the original audio sample and Sbackground is the signal of the background

scene. Worig and Wbackground are weighting volume parameters determined under the

premise that Worig + Wbackground = 1. For each Sorig audio, two or three Smix audio files with different weights were generated, depending on the dataset.

* ***Norm***: Normalizes an audio file to a dB level. This function gets the requested ‘db\_level’ as a parameter.

DAS can create a large number of synthetic samples from original “clean” samples. The number of synthetic samples is determined by the configuration of the methods to use (tempo, pitch, and background noise are the four supported methods now, but any other method can be easily added), and by configuration of each method value. For example, if the configuration for the pitch method contains four different values for the semitone parameter, then the system will generate four different augmented audio samples with the desired pitch level. A significant advantage in DAS is the fact that different effects can be combined for each recording. This allows us to control the size and the diversity of the dataset we want to create; e.g., if the desired methods are tempo (with factors 0.9 and 1.1) and pitch (with semitones 2 and -2), an additional eight audio samples will be created from one original audio (two audios with only tempo effect, two with only pitch and four that are result of combining tempo and pitch parameters together).

In addition, DAS can generate white noise (Auditok) in the background of each original audio file. This functionality was added because white noise is produced by combining sounds of all different frequencies together and can be used to mask other sounds, such as background noises.

*Network architecture*

*FNN*

The implementation was carried out in TensorFlow. A set of three layers was trained with 280 units and a tangent activation function, and 290 and 300 units for the second and third layers, respectively, with a sigmoid function. Since we are dealing with a multiclass classification problem, the output layer used SoftMax as its activation function, which outputs a vector that represents the probability distributions of a list of potential outcomes. The loss function used was multi-class cross-entropy. Training was done using the Adam optimizer with an initial learning rate of 0.009 and ~300 epochs. Every five epochs, we checked the F-score performance on the validation set and saved the model with the best validation F-score.

**Figure 2 around here**

*LSTM*

The implementation was carried in TensorFlow. A set of two LSTM layers with 50 units each was used. The output used SoftMax as its activation function and multi-class cross-entropy as the loss function. Training was done using the Adam optimizer with an initial learning rate of 0.008 and ~300 epochs, and during each epoch a mini-batch of approximately 10% of the total length was trained. Every five epochs, we checked the F-score performance on the validation set and saved the model with the best validation F-score.

**Figure 3 around here**

Experimental Setup

*Creating Different Datasets using DAS*

It is important to choose the augmentation parameters such that the semantic validity of the label is maintained. To create our augmentation sets, we chose several parameters: for tempo, a factor in the range of 0.86-1.14; for pitch, a semitone in the range of (-2.4)-2.4; and for background noises, Worig in the range of 0.83-0.93 and Wbackground in the range of 0.07-0.17. We claim that each augmentation is helpful, but their combination gives better results. The resulting augmentation sets are described in the next table.

**Table 3 around here**



For each speaker, 120 different audio samples were used from our original dataset (600 files altogether from four interjection folders and one folder of negative examples), whose overall length is about eight minutes, to generate seven augmentation sets with a total of 180,000 audio samples, whose overall length was about **78 hours**. For all speakers, the system generated **more than 300 hours** of augmentation data from 32 minutes of original recordings. Based on the results obtained in the experiments, at a more advanced stage, an additional function was needed. The ‘Norm’ method was added to DAS and two additional datasets (datasets 8 and 9) were created, described later in this paper.

*Scenario*

In this section we describe scenarios using different configurations, while training and test sets have been kept separate. The F-score, which is a commonly used measure of classification accuracy that gives equal weight to precision (how many instances were correctly predicted, given all the predicted labels for a given class X), and recall (how many instances were correctly captured, from all instances that should have a label X) were used for computing the score of the recognition process. The F-Score equation is shown below:

(2)

Where TP (true positive) is the number of correct classifications by the classifier and FN (false negative) is the number of misclassified predictions, where the model incorrectly predicts it is not from label X. FP (false positive) is the number of misclassified predictions where the model incorrectly predicts it is from label X. The choice was based on the intuition that a good classifier should maximize both precision and recall simultaneously. So a model with good precision and recall will score better than a model that has extremely good performance on just one of them.

We evaluate the performance of our models on scenarios that test previously unseen speakers. In each scenario, the model is trained with each of the data augmentation sets described in Table 3, while the F-score for each data set is computed by the mean F-score of seven runs.

### *Scenario description*

The goal is to check how augmentation improves the results for the previously unseen speaker. The first step of this scenario was to train the model with clean unsynthesized data from two speakers (one female and one male) and then validate and test it separately on clean unsynthesized data from two unseen speakers (one female and one male). Next, we trained it separately on each of our seven data augmentation sets, and then validated and tested it the same way as in the first step, i.e., on the same unsynthesized data from each unseen speaker.

We extended this scenario for the LSTM model and trained the model not only with two speakers, but also with one and three different speakers. The testing was done separately on unsynthesized data of the same two unseen speakers. Based on the results obtained for the experiments described above, another extension and important test were done with only one recording of ten speakers. Ten different speakers of both genders (ages 11 to 75) were recorded only once for each word. One new dataset augmented by tempo, pitch, and background methods was created from those ten speaker’s audio recordings to train the model, and again as in the previous tests, the testing was done separately on unsynthesized data of the same two unseen speakers.

Results and Discussion

The experiment was conducted in our study for each model (FNN and LSTM). The accuracy assessment was measured by comparing the baseline (unsynthesized, clean dataset) F-score with the F-score of each of the proposed augmentation datasets. First, the model was trained by two speakers and better results were obtained by the LSTM model. For this reason, it was decided to extend the experiment in this model to train it with one and three different speakers as well. In doing so, we wished to examine the impact of the number of trained speakers on the F-score.

*FNN Results*

Results of the FNN model are presented in the tables below. The model was trained twice with samples of two speakers (speaker A and speaker B). It was first validated on unseen speaker C and tested on speakers D and E separately (Table 4), then validated on unseen speaker D and tested on speakers C and E separately (Table 5). We can see that almost every one of the augmented methods significantly improves the F-score relative to the baseline F-score that trained with original unprocessed samples. Except for a few cases, each of the augmentation sets were helpful in this scenario, and in four of six columns, the highest classification F-score improvement for each unseen speaker was achieved by the dataset combined with at least two augmentation methods.

**Table 4 around here**



As seen in Table 5, the best F-score for speakers C and E was achieved by one method, but not far behind, the second highest F-score was achieved by a combination of at least two methods. The F-score of unseen speaker E was very low relative to the F-score achieved for tested speakers D and C. Still, all augmented datasets significantly improved the F-score.

**Table 5 around here**



*LSTM Results*

Results of the LSTM model are presented in the subsequent tables. Tables 6 and 7 present the results where the model trained with audio samples of one speaker (speaker A). Tables 8 and 9 present the results where the model trained with audio samples of two speakers (speaker A and B), and Table 10 presents the results where the model trained with audio samples of three speakers. After training, models were first validated on previously unseen speaker C and tested on speakers D and E, and additionally validated on previously unseen speaker D and tested on speakers C and E (except for the case where the model trained on three speakers and tested only on speaker E).

*Training By One Speaker*

In Table 6, each of the augmented methods significantly improves the validation F-score relatively to the baseline F-score that trained with original, unprocessed samples. The test F-score of speaker D also achieved significant improvement except for the pitch method, and the test F-score of speaker E achieved lower results than speaker D and improvement only in some cases, but for all speakers the best result was achieved by a combination of methods (22.3% and 23.2% with Tempo+Pitch for speakers D and E, and 21% improvement with three methods for speaker C).

**Table 6 around here**



In Table 7, most of the methods improve the F-score for speakers C and D (validated), and improvements for speaker E are only seen in three augmented datasets. Unlike Table 6, the best improvement for all speakers was achieved by only one method (pitch), but combined datasets for speaker C and D also achieved very significant improvement. A possible explanation as to why pitch is such an important factor in Table 7 is that the model trained only with one speaker and was validated by speaker with very different characteristics (44 year-old male and 81 year-old female).

**Table 7 around here**



*Training By Two Speakers*

Results in Tables 8 and 9 are analogous, in terms of trained speakers, to the results of the FNN model in Tables 4 and 5. The column “FNN” presents increase/decrease compared to the results of the same dataset in Tables 4 and 5. As shown in Table 8, the greatest improvement was achieved by combining all three augmentation methods for all unseen speakers C, D, and E. Each of the augmented datasets (except a small decrease for the Tempo + BckGrd dataset for tested speaker D) had a significantly improved F-score relative to the baseline F-score, and exceeded the results of the corresponding dataset shown in Table 4. The best validation F-score achieved in LSTM for speaker C was 0.631, which is equal to the best validation F-score achieved for speaker C in FNN, but the F-score of tested speakers D and E was higher compared to the analogous FNN F-score. The efficiency of DAS is noticeable when looking at the results of speaker D. The F-score of Speaker D with an unsynthesized dataset in LSTM was 5.2% lower than the corresponding F-score in FNN, but six of seven augmented datasets in LSTM produce a much higher F-score in LSTM, corresponding to the analogous F-score in the FNN model.

**Table 8 around here**



In Table 9, all augmented datasets improve the F-score for speakers C and D (except a small decrease for the Tempo + BckGrd dataset for validating speaker D) and improvements for speaker E were achieved only in two augmented datasets. The highest classification F-score improvement for unseen speakers C and D was achieved by the dataset combined with all three augmentation methods (36.3% for speaker C and 22.6% for speaker D), while the best F-score improvement for speaker E was achieved by the pitch dataset. The F-score in this table shows a consistent growth of the LSTM F-score for speakers D and E compared to the F-score of corresponding speakers in FNN. For speaker C the results are slightly different, while four of seven augmented datasets, including the dataset with the best F-score, achieved a lower result compared to the baseline dataset.

**Table 9 around here**



Because speaker E had poorer results relative to the baseline dataset results in Tables 6,7, and 9, and relative to the results of tested speakers C and D, we sought to investigate the reasons. From examining several audio sample waveforms of speakers A to E, we discerned that the amplitude level varies greatly between speakers as seen in Figure 5, even though all speakers recorded in the same conditions and environment. All the models were trained on speakers A-D, validated on speakers C or D, and tested on speakers C, D and E. Speaker E was the only speaker that functioned solely as a tested speaker, and he was tested on models that trained and validated on speakers with a much lower amplitude level. In order to improve the results for speaker E, the goal was to find a way to adjust the training dataset for better classification of speakers with different amplitude levels. The “norm” method added to DAS was intended to make this adjustment by normalizing an audio sample to a particular dB (decibel) level.

Dataset 8 from Table 3 was created to examine the effect of the ‘norm” method on the results described in Tables 8 and 9, and 5 dB level values (-1,0,1,2,3) were tried in this dataset. As seen in Tables 8 and 9, the F-score of speaker E with the new dataset including the “norm” method outperforms the best F-score achieved without the norm method (0.476 instead of 0.401 in Table 8, and 0.506 instead of 0.457 in Table 9). For speakers C and D in both tables, the “norm” dataset achieved an F-score that generally was close to or even exceeded the best F-score without “norm”. Adding this new method illustrates the effectiveness of DAS: it can be adjusted and easily made more efficient by adding new methods and applying different characteristics to the dataset.

*Training By Three Speakers*

The third phase of this scenario, where the model trained with three speakers and was tested with an unseen speaker, supports the claim that augmentation is helpful for improving the model F-score. For validation sets, the combination of the three methods achieved the highest F-score. Speaker C achieved a 23.1% improvement in F-score (0.751) over the F-score of the basic unsynthesized dataset, and speaker D achieved an improvement of 13%. Testing the models was done only on speaker E, and although the F-score was much lower than the validation F-score, still, DAS managed to greatly improve the F-score relative to the baseline dataset F-score. As in the case of training the model with two speakers, the assumption is that the F-score of speaker E can be improved by using a different configuration that includes the “norm” method. Also of note is the improvement of test F-score compared to the corresponding F-score achieved in Tables 8 and 9, where the model trained with two speakers.

**Table 10 around here**



The F-Score gives some perspective on the quality of the model, but its main problem is that it hides the detail we need to better understand the performance of our classification model. For example, we may get a classification F-score of 85%, but we do not know if that is because all classes are being predicted equally well or whether one or two classes are being neglected by the model. By knowing the true labels, we can use another important metric – a confusion matrix, where each column represents the number of instances in a predicted class and each row represents the instances in an actual class. This metric helps to determine where the system is becoming confused between two classes by comparing the predicted classes with the actual classes.

Figure 6 shows two pairs, one pair for each speaker C and D, of confusion matrices taken from training with three speakers in the first experiment. Each pair includes one matrix trained by original unsynthesized data, and another matrix trained by all three methods. Table 11 presents the recall and precision values for each matrix.

For speaker C, labels 0 (“negative example”) and 3 (“nah”) were poorly predicted with unsynthesized data. Recall of label 3 is 0.296, i.e., from all labels of class 3, only 45 instances (29.6%) were correctly captured (64.5%, which are 98 instances predicted as negative examples, 6 predicted as “oy”, and 3 predicted as “mmm”). The precision of label 3 is 0.455, i.e., from 99 instances that predicted as “nah”, only 45 instances are really from that class (52 belong to the “ahah” class and 2 are negative examples). With synthesized data, recall and precision significantly improved to 0.983 and 0.871 respectively. Recall of label 0 is 0.418, i.e., from all labels of class 0, only 79 instances (41.8%) were correctly captured (51.9%, which are 98 predicted as “oy”). The precision of label 0 is 0.403, i.e., from 196 instances that predicted as negative examples, only 79 instances are really from that class (98 belong to the “nah” class and 15 to “oy”). With synthesized data, recall and precision improved to 0.57 and 0.643 respectively. In the same way the recall of label 1 (“ahah”) was improved from 0.566 to 0.81. Except for the recall of label 4 (“oy”), which significantly decreased from 0.877 to 0.583, all other values have not changed significantly.

For speaker D, a significant improvement between the datasets was detected in the precision of label 0 (negative examples); 45.9% with unsynthesized data increases to 0.678 in synthesized data and recall of 0.463 for class 1 (“ahah”), and 0.39 for class 2 (“mmm”) increases to 0.612 and 0.917 respectively. However, a decrease was detected in the recall of negative examples, from 0.753 to 0.534, and in the precision of class 2 (“mmm”) from 0.93 to 0.79.

**Table 11 around here**



DAS greatly improves recall and precision of the trained model compared to the trained model with unsynthesized data. Still, most of the higher values outside the diagonal line belong to the row or column of negative examples, which means that many samples belonging to the negative class incorrectly captured this class and many samples predicted as belonging to the negative class actually belong to another class. To improve that, we suggest increasing the number of samples in the negative class, especially with label samples of words that are similar to interjection words from other classes.

**Figure 6 around here**

The second issue examined is the impact of the number of trained speakers on the F-score of the previously unseen speaker. In each case of training with one, two, and three speakers, the average validation F-score of speakers C and D was calculated among all eight datasets (one original unsynthesized dataset and seven augmented datasets). Figure 7 shows the average of each dataset compared among the amounts of speakers. Improvement can be noticed as the number of trained speakers increases. The F-score of all eight datasets trained on three speakers is larger than the corresponding dataset trained on two speakers, and the F-score of 7 from 8 datasets trained on two speakers is larger than the corresponding dataset trained on one speaker. The claim that the combination of three methods achieved the higher improvement of F-score relative to only unsynthesized data is also supported by this figure.

*Training By Ten Speakers*

The strong effect of the number of trained speakers on the F-score of previously unseen speakers convinced us to do one test with a dataset created from only one unsynthesized audio sample for each word. The unsynthesized audio file was recorded among ten different speakers, as mentioned in the scenario description. In most cases, validation and testing in the previous experiments with 1, 2, and 3 speakers yielded the best results by combining the three effects (tempo, pitch, and background). Initially, this test was made with a new dataset created from ten speaker’s audio samples with the same parameters of the corresponding dataset created for 1, 2, and 3 speakers. The first line in Tables 12 and 13 shows the results of this experiment. A great improvement can be seen in validation F-scores and test F-scores for speakers C and D, but on the other side, the test F-score on speaker E was much lower from that in Table 10. where the model trained with an audio sample of three speakers.

To improve the model, we built a new dataset including all four effects provided by DAS (dataset 9 from Table 3). Table 12 shows a great improvement both for speakers D and E, and in Table 13, the F-score of speaker C decreased a bit to 0.665, but a notable improvement was achieved for speaker E.

**Table 12 around here**

**Table 13 immediately after**







There are two important findings of this test. It confirms and supports the claim that the more the model is trained with additional speakers, the further its F-score increases Second, our data augmentation system is an essential and critical tool because it allows for better results with minimal data, and for training data, as in the case of speaker E.

**Figure 7 around here**

Figure 7 shows confusing matrices of two tests for speaker E: one for testing on dataset 7 without the “norm” method, and the other for testing on dataset 9 with the “norm” method. As shown in Figure 8, the main problem is in classifying the negative example word. The precision of label 0 without the “norm” method is 0.209, i.e., from all instances that were predicted as negative, only 21% of instances are really from that class. The “norm” method partially fixed it while increasing precision to 0.52. As a result, classification of classes 1 and 2 were improved, but still, many samples belonging to class 3 were incorrectly predicted as belonging to class 0 without the “norm” effect, and were incorrectly predicted as belonging to class 2 with the “norm” effect. We believe that better DAS configuration can offer further improvement.

Finally, the models seem to be more robust when using a combination of at least two effects. More precisely, in 7 of 9 cases the combination of three methods achieved a higher improvement in F-score compared to using only original unsynthesized data. This implies that using varied settings for each effect can improve the results beyond what we obtained in the scenarios described.

In future experiments, we will add a preprocessing phase for selecting augmentation hyperparameters, which will create effects and combinations of effects with several different settings. Each setting will generate a new dataset that will be trained on the model. This preprocessing can be helpful in selecting the data augmentation set that is most effective for the appropriate scenario. Another extension can be adding a new augmentation method beyond the four methods mentioned in our work.

Conclusions

In this work we proposed a new problem definition for interjection classification implemented using two network architectures: a fully connected feedforward neural network (FNN) and Long Short-Term Memory (LSTM), which in combination with a set of audio data augmentations, produces state-of-the-art results for interjection classification. We showed that the improved performance stems from the combination of a basic classification model and an augmented training set. This combination outperformed the proposed architectures with no augmentation at all.

We conducted an experiment to explore the influence of different sets of four data augmentation methods on unsynthesized audio samples. We observed that combining the augmentation methods gives better results than each alone. As shown in Table 14, in validation sets, 7 of 8 cases achieved the best F-Score by combining three methods, and 8 of 14 cases in test sets achieved the best F-Score by combining at least two methods. Three of four cases where the best F-Score was achieved by one method were from tested speaker E. In one of these three cases the best F-Score was achieved by the pitch method (Table 9). We saw in that case that combining the norm method with the pitch method improved the top F-Score. We suggest that the performance of the model could be improved further by applying new functionality for better selection of more appropriate data augmentation sets for the desired scenario.



**Table 14 around here**

As part of the experiment, we showed the positive impact of the number of trained speakers on the F-score of previously unseen speakers by training the LSTM model with one, two, and three speakers. This positive impact, in addition to the importance of data augmentation, received confirmation after the second extension of the second scenario in which the LSTM model trained with a new dataset made by ten different speakers recorded only once for each word.

References

Chandra, E., Senthildevi, K.A.. (2015). Keyword Spotting: An Audio Mining Technique in Speech Processing – A Survey, IOSR Journal of VLSI and Signal Processing (IOSR-JVSP).‏

Chen, G., Parada, C., & Heigold, G. (2014, May). Small-footprint keyword spotting using deep neural networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP) (pp. 4087-4091). IEEE.‏

Cohn, M., Chen, C. Y., & Yu, Z. (2019, September). A Large-Scale User Study of an Alexa Prize Chatbot: Effect of TTS Dynamism on Perceived Quality of Social Dialog. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue* (pp. 293-306).‏

Eronen, A. J., Peltonen, V. T., Tuomi, J. T., Klapuri, A. P., Fagerlund, S., Sorsa, T., ... & Huopaniemi, J. (2005). Audio-based context recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1), 321-329.‏

Goddard, C. (2014). Interjections and emotion (with special reference to “surprise” and “disgust”). *Emotion Review*, 6(1), 53-63.‏

Gouda, S. K., Kanetkar, S., Harrison, D., & Warmuth, M. K. (2018). Speech Recognition: Keyword Spotting Through Image Recognition. *arXiv preprint* arXiv:1803.03759.‏

Hoy, M. B. (2018). Alexa, Siri, Cortana, and more: an introduction to voice assistants. *Medical reference services quarterly*, 37(1), 81-88.‏

Lopatovska, I., Rink, K., Knight, I., Raines, K., Cosenza, K., Williams, H., ... & Martinez, A. (2019). Talk to me: Exploring user interactions with the Amazon Alexa. *Journal of Librarianship and Information Science*, 51(4), 984-997.‏

Majeed, S. A., Husain, H., Samad, S. A., & Idbeaa, T. F. (2015). MEL FREQUENCY CEPSTRAL COEFFICIENTS (MFCC) FEATURE EXTRACTION ENHANCEMENT IN THE APPLICATION OF SPEECH RECOGNITION: A COMPARISON STUDY. *Journal of Theoretical & Applied Information Technology*, 79(1).‏

Pradhan, A. (2012). Support vector machine-A survey. *International Journal of Emerging Technology and Advanced Engineering*, 2(8), 82-85.‏

Proutskova, P., Rhodes, C., Wiggins, G., & Crawford, T. (2012). Breathy or resonant-A controlled and curated dataset for phonation mode detection in singing.‏

Zhou, Y., Xiong, C., & Socher, R. (2017). Improved regularization techniques for end-to-end speech recognition. *arXiv preprint* arXiv:1712.07108.‏

Table 1. Four selected interjections that recorded into separate sound clips.

|  |  |  |
| --- | --- | --- |
| interjection | alternate/similar | translation/meaning |
| nah |  | **"No"** - Informal no |
| mmm | mhm, uh-hu | **"Yes"** - Agreement, acknowledgement |
| ahah | Aha, ahh | **"I understand"** - Understanding,  Confirmation |
| oy | oy vey | **"Oh no..."** - Mainly Jewish: Used to express self-pity, or expression of  unexpected situation |

Table 2. Speaker profiles and number of audio samples recorded for each speaker and word.

|  |  |  |
| --- | --- | --- |
| speaker | number of audio samples | profile |
| A | 850~ | Gender: **Male**, Age:**44**, Native language: **Hebrew** |
| B | 550~ | Gender: **Female**, Age:**42**, Native language: **Hebrew** |
| C | 300~ | Gender: **Male**, Age:**16**, Native language: **Hebrew** |
| D | 300~ | Gender: **Female**, Age:**81**, Native language: **Spanish** |
| E | 300~ | Gender: **Male**, Age:**50**, Native language: **Hebrew** |

Table 3. Description of the different augmentation sets created.

|  |  |  |  |
| --- | --- | --- | --- |
|  | original samples per class | used effects | generated samples per class and per speaker |
| 1 | 120 | Pitch | 2520 |
| 2 | 120 | Tempo | 2520 |
| 3 | 100 | Background | 2900 |
| 4 | 60 | Tempo + Pitch | 4860 |
| 5 | 40 | Pitch + Background | 6880 |
| 6 | 40 | Tempo + Background | 6880 |
| 7 | 10 | Tempo + Pitch + Background | 9320 |
| 8 | 50 | Pitch + Norm | 1500 |
| 9 | 1 | Tempo + Pitch + Norm + Background | 932 |

Table 4. Results of the first scenario for the FNN model, validated on unseen speaker C and tested on speakers D and E. The percentages within the parentheses present the F-score increase/decrease compared to the baseline, original, unsynthesized dataset F-score.

|  |  |  |  |
| --- | --- | --- | --- |
| **FNN:** train with 2 speakers and validate with unseen speaker c | | | |
| training dataset  (speakers a+b) | validate f-score (**speaker c**) | test f-score (**speaker d**) | test f-score (**speaker e**) |
| Unsynthesized Data | 0.42 | 0.443 | 0.208 |
| Pitch | 0.516 (↑ 22.9%) | 0.425 (↓ 4.2%) | 0. 24 (↑ 15.4%) |
| Tempo | 0.485 (↑ 15.5%) | 0.45 (↑ 1.6%) | 0. 252 (↑ 21.2%) |
| Background | 0.504 (↑ 20%) | 0.401 (↓ 10.5%) | 0.3 (↑ 44.2%) |
| Tempo + Pitch | 0.538 (↑ 28.1%) | 0.425 (↓ 4.2%) | 0.25 (↑ 20.2%) |
| Pitch + Background | 0.544 (↑ 29.5%) | 0.445 (↑ 0.4%) | 0.247 (↑ 18.8%) |
| Tempo + Background | 0.521 (↑ 24%) | 0.447 (↑ 0.9%) | **0.304** (↑ 46.2%) |
| Tempo + Pitch + Background | **0.631** (↑ 50.2%) | **0.503** (↑ 13.5%) | 0.23 (↑ 10.6%) |

Table 5. Results of the first scenario for the FNN model with validation performed on unseen speaker D and testing performed on speaker C and E. The percentages within the parentheses present the F-score increase/decrease compared to the baseline original unsynthesized dataset F-score.

|  |  |  |  |
| --- | --- | --- | --- |
| **FNN:** train with 2 speakers and validate with unseen speaker d | | | |
| training dataset  (speakers a+b) | validate f-score (**speaker d**) | test f-score (**speaker c**) | test f-score (**speaker e**) |
| Unsynthesized Data | 0.521 | 0.333 | 0.213 |
| Pitch | 0.525 (↑ 0.8%) | 0.455 (↑ 36.6%) | 0. 266 (↑ 24.9%) |
| Tempo | 0.521 (0%) | **0.528** (↑ 58.6%) | 0. 247 (↑ 16%) |
| Background | 0.506 (↓ 3%) | 0.466 (↑ 40%) | **0.298** (↑ 40%) |
| Tempo + Pitch | 0.534 (↑ 2.5%) | 0.505 (↑ 51.7%) | 0.282 (↑ 32.4%) |
| Pitch + Background | 0.497 (↓ 4.8%) | 0.404 (↑ 21.3%) | 0.254 (↑ 19.2%) |
| Tempo + Background | 0.539 (↑ 3.5%) | 0.38 (↑ 14.1%) | 0.282 (↑ 32.4%) |
| Tempo + Pitch + Background | **0.591** (↑ 13.4%) | 0.525(↑ 57.7%) | 0.274 (↑ 28.6%) |

Table 6. Results of the first scenario for the LSTM model where validation was performed on unseen speaker C and testing was performed on speaker D and E. The percentages within the parentheses present the F-score increase/decrease compared to the baseline original unsynthesized dataset F-score.

|  |  |  |  |
| --- | --- | --- | --- |
| **lstm:** train with 1 speaker and validate with unseen speaker c | | | |
| training dataset  (speakers a) | validate  f-score (**speaker c**) | test  f-score (**speaker d**) | test  f-score  (**speaker e**) |
| Unsynthesized Data | 0.501 | 0.403 | 0.297 |
| Pitch | 0.603 (↑ 20.4%) | 0.454 (↑ 12.7%) | 0. 365 (↑ 22.9%) |
| Tempo | 0.503 (↑ 0.4%) | 0.392 (↓ 2.8%) | 0. 302 (↑ 1.7%) |
| Background | 0.54 (↑ 7.8%) | 0.431 (↑ 6.9%) | 0.249 (↓ 19.3%) |
| Tempo + Pitch | 0.595 (↑ 18.8%) | 0.424 (↑ 5.2%) | 0.265 (↓ 12.1%) |
| Pitch + Background | 0.58 (↑15.8%) | **0.493** (↑ 22.3%) | **0.366** (↑ 23.2%) |
| Tempo + Background | 0.562 (↑ 12.2%) | 0.419 (↑ 4%) | 0.275 (↓ 8%) |
| Tempo + Pitch + Background | **0.606** (↑ 21%) | 0.478(↑ 18.6%) | 0.296 (↓ 0.3%) |

Table 7. Results of the first scenario for the LSTM model where validation is performed on unseen speaker D and testing is performed on speaker C and E. The percentages within the parentheses present the F-score increase/decrease compared to the baseline original unsynthesized dataset F-score.

|  |  |  |  |
| --- | --- | --- | --- |
| **lstm:** train with 1 speaker and validate with unseen speaker d | | | |
| training dataset  (speaker a) | validate  f-score (**speaker d**) | test  f-score (**speaker c**) | test  f-score  (**speaker e**) |
| Unsynthesized Data | 0.494 | 0.378 | 0.351 |
| Pitch | **0.576** (↑ 16.6%) | **0.58** (↑ 53.4%) | **0. 443** (↑ 26.2%) |
| Tempo | 0.493 (↓ 0.2%) | 0.411 (↑ 8.7%) | 0. 323 (↓ 8.7%) |
| Background | 0.502 (↑ 1.6%) | 0.448 (↑ 18.5%) | 0.235 (↓ 49.4%) |
| Tempo + Pitch | 0.537 (↑ 8.7%) | 0.552 (↑ 46%) | 0.389 (↑ 10.8%) |
| Pitch + Background | 0.57 (↑ 15.4%) | 0.536 (↑ 41.8%) | 0.341 (↓ 2.9%) |
| Tempo + Background | 0.493 (↓ 0.2%) | 0.498 (↑ 31.7%) | 0.228 (↓ 53.9%) |
| Tempo + Pitch + Background | 0.559 (↑ 13.2%) | 0.533(↑ 41%) | 0.35 (↓ 0.3%) |

Table 8. Results of first scenario for LSTM where validation is performed on unseen speaker C and testing is performed on speakers D and E. The percentages within the parentheses present increase/decrease compared to the baseline clean dataset F-score. The “**FNN**” columns present improvement/decrease compared to the results of the same dataset in Table 4. (**BckGrd** = Background)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **lstm:** train with 2 speakers and validate with unseen speaker c | | | | | | |
| training dataset  (speakers a+b) | validate  f-score  (**speaker c**) | fnn | test  f-score  (**speaker d**) | fnn | test  f-score  (**speaker e**) | fnn |
| Unsynthesized Data | 0.516 | ↑22.9% | 0.421 | ↓5.2% | 0.279 | ↑34.1% |
| Pitch | 0.573 (↑11%) | ↑11% | 0.492 (↑9.2%) | ↑15.8 | 0.381 (↑36.6%) | ↑58.8% |
| Tempo | 0.568 (↑10.1%) | ↑17.1% | 0.481 (↑4.9%) | ↑6.9% | 0.382 (↑36.9%) | ↑51.6% |
| BckGrd | 0.593 (↑14.9%) | ↑17.7% | 0.452 (↑4.7%) | ↑12.7% | 0.323 (↑15.8%) | ↑7.7% |
| Tempo + Pitch | 0.575 (↑11.4%) | ↑6.9% | 0.521 (↑6.5%) | ↑22.6% | 0.398 (↑42.7%) | ↑59.2% |
| Pitch + BckGrd | 0.588 (↑14%) | ↑8.1% | 0.528 (↑10.3%) | ↑18.7% | 0.372 (↑33.3%) | ↑50.6% |
| Tempo + BckGrd | 0.526 (↑1.9%) | ↑1% | 0.432 (↓0.2%) | ↓3.5% | 0.326 (↑16.8%) | ↑7.2% |
| Tempo + Pitch + BckGrd | **0.631** (↑22.3%) | 0% | **0.566** (↑22.6%) | ↑12.5% | **0.401** (↑43.7%) | ↑74.3% |
| Pitch + Norm | 0.59(↑14.3%) |  | 0.557 (↑32.3%) |  | **0.476** (↑70.6%) |  |

Table 9**.** Results of first scenario for LSTM where validation is performed on unseen speaker D and testing performed on speaker C and E. The percentages within the parentheses present increase/decrease compared to the baseline clean dataset F-score. The “**FNN**” columns present increase/decrease compared to the results of the same dataset in Table 5. (**BckGrd** = Background)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **lstm:** train with 2 speakers and validate with unseen speaker c | | | | | | | |
| training dataset  (speakers a+b) | validate  f-score  (**speaker d**) | fnn | test  f-score  (**speaker c**) | fnn | test  f-score  (**speaker e**) | fnn |
| Unsynthesized Data | 0.535 | ↑2.7% | 0.372 | ↑1.2 | 0.426 | ↑100% |
| Pitch | 0.584 (↑9.2%) | ↑11.2% | 0.481 (↑29.3%) | ↑5.7 | **0.457** (↑7.3%) | ↑71.8% |
| Tempo | 0.561 (↑4.9%) | ↑7.7% | 0.382 (↑2.7%) | ↓38.2% | 0.37 (↓15.1%) | ↑49.8% |
| BckGrd | 0.56 (↑4.7%) | ↑10.7% | 0.429 (↑15.3%) | ↓8.6% | 0.343 (↓24.2%) | ↑15.1% |
| Tempo + Pitch | 0.57 (↑6.5%) | ↑6.7% | 0.449 (↑20.7%) | ↓12.5% | 0.452 (↑6.1%) | ↑60.3% |
| Pitch + BckGrd | 0.59 (↑10.3%) | ↑18.7% | 0.488 (↑31.2%) | ↑20.8% | 0.397 (↓7.3%) | ↑56.3% |
| Tempo + BckGrd | 0.534 (↓0.2%) | ↓0.9% | 0.43 (↑15.6%) | ↑13.2% | 0.385 (↓10.6%) | ↑36.5% |
| Tempo + Pitch + BckGrd | **0.656** (↑22.6%) | ↑11% | **0.507** (↑36.3%) | ↓3.6% | 0.396 (↓7.6%) | ↑44.5% |
| Pitch + Norm | 0.589(↑10.1%) |  | **0.533** (↑43.3%) |  | **0.506** (↑18.8%) |  |

Table 10. Results of the first scenario for the LSTM model. The percentages within the parentheses present increase/decrease compared to the baseline clean dataset F-score.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **lstm:** training with 3 speakers and validating with 2 unseen speakers | | | | |
|  | training (speakers **a + b + d**) | | training (speakers **a + b + c**) | |
| dataset | validate  f-score  (**speaker c**) | test f-score  (**speaker e**) | validate  f-score  (**speaker d**) | test f-score  (**speaker e**) |
| Unsynthesized Data | 0.61 | 0.425 | 0.592 | 0.446 |
| Pitch | 0.712 (↑16.7%) | 0.506 (↑19.1%) | 0.636 (↑7.4%) | 0.481 (↑7.8%) |
| Tempo | 0.606 (↓0.6%) | 0.48 (↑12.9%) | 0.625 (↑5.6%) | **0.549** (↑23.1%) |
| Background | 0.609 (↓0.2%) | 0.448 (↑5.4%) | 0.568 (↓4.2%) | 0.408 (↓9.3%) |
| Tempo+Pitch | 0.666 (↑9.2%) | **0.543** (↑27.8%) | 0.619 (↑4.6%) | 0.515 (↑15.5%) |
| Pitch+Background | 0.67 (↑9.8%) | 0.458 (↑7.8%) | 0.636 (↑7.4%) | 0.422 (↓5.7%) |
| Tempo+Background | 0.6 (↓1.7%) | 0.406 (↓4.7%) | 0.548 (↓8%) | 0.434 (↓2.8%) |
| Tempo+Pitch+Background | **0.751** (↑23.1%) | 0.377 (↓12.7%) | **0.669** (↑13%) | 0.418 (↓6.7%) |

Table 11. Recall and precision values for matrices shown in Figure 2.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | speaker c | | | | speaker d | | | |
|  | unsynthesized | | synthesized | | unsynthesized | | synthesized | |
| class | recall | precision | recall | precision | recall | precision | recall | precision |
| negative | 0.418 | 0.403 | 0.57 | 0.643 | 0.753 | 0.459 | 0.534 | 0.678 |
| ahah | 0.566 | 0.925 | 0.81 | 0.967 | 0.463 | 0.747 | 0.612 | 0.775 |
| mmm | 0.968 | 0.932 | 0.984 | 0.824 | 0.39 | 0.93 | 0.917 | 0.79 |
| nah | 0.296 | 0.455 | 0.983 | 0.871 | 0.56 | 0.471 | 0.602 | 0.514 |
| oy | 0.877 | 0.546 | 0.583 | 0.589 | 0.75 | 0.665 | 0.743 | 0.614 |

Table 12**.** Results of first scenario for LSTM trained on ten speakers where validation was performed on unseen speaker C and testing was performed on speakers D and E. The percentages within the parentheses present the increase/decrease compared to the test done with three speakers.

|  |  |  |  |
| --- | --- | --- | --- |
| **lstm:** train with 10 speakers and validate with unseen speaker c | | | |
| training dataset  (10 speakers) | validate f-score (**speaker c**) | test f-score  (**speaker d**) | test f-score  (**speaker e**) |
| Tempo + Pitch + Background | 0.805 (↑ 7.2%) | 0.7 | 0.463 (↓ 17.3%) |
| Tempo + Pitch + Norm +Background | 0.695 | **0.742** | **0.543** |

Table 13. Results of first scenario for LSTM trained on ten speakers where validation was performed on unseen speaker D and testing was performed on speaker C and E. The percentages within the parentheses present the increase/decrease compared to the test done with three speakers.

|  |  |  |  |
| --- | --- | --- | --- |
| **lstm:** train with 10 speakers and validate with unseen speaker d | | | |
| training dataset  (10 speakers) | validate f-score (**speaker d**) | test f-score  (**speaker c**) | test f-score  (**speaker e**) |
| Tempo + Pitch + Background | 0.781 (↑ 16.7%) | 0.681 | 0.464 (↓ 18.3%) |
| Tempo + Pitch + Norm +Background | 0.765 | 0.665 | **0.532** |

Table 14. Number of effects used that achieved the higher F-Score.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| validate set | | | | test set | | | |
| 1 effect | 2 effects | 3 effects | 1 effect | | 2 effects | 3 effects |
| 0 of 2 | 0 of 2 | 2 of 2 | 2 of 4 | | 1 of 4 | 1 of 4 |
| 1 of 6 | 0 of 6 | 5 of 6 | 4 of 10 | | 3 of 10 | 3 of 10 |

Figure 1. Automatic speech recognition system diagram.

Figure 2. Feedforward neural network architecture.

Figure 3. LSTM architecture. x<t> is the input at time t. y<t> is the input at time t. a[l]<t> is the input of cell at time t+1 from layer l.

Figure 4. Unsynthesized waveforms of speakers A-E saying “ahah”.

Figure 5. Top left matrix: trained by unsynthesized data - validation F-score for speaker C - 0.621. Top right matrix: trained by synthesized data - validation F-score for speaker C: 0.771. Bottom left matrix: trained by unsynthesized data - validation F-score for speaker D - 0.57. Bottom right matrix: trained by synthesized data - validation F-score for speaker D: 0.677.

Figure 6. The average F-score of both speakers C and D according to the number of trained speakers. The dark blue bar indicates the average F-score of datasets trained on 3 speakers. Beneath it, the blue bar indicates the average F-score of datasets trained on 2 speakers, and the orange bar indicates the average F-score of datasets trained on 1 speaker.

Figure 7. Left: Trained by synthesized dataset 7 with 10 speakers - Test F-score for speaker E - 0.45. Right: Trained by synthesized dataset 9 with 10 speakers - Test F-score for speaker E: 0.543.

Diagram

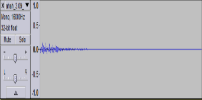
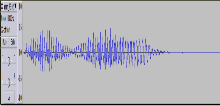
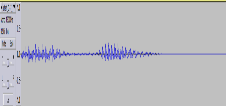
Description automatically generated

Diagram

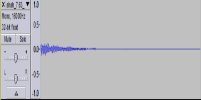
Description automatically generated

Diagram

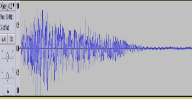
Description automatically generated



**Speaker C**



**Speaker D**



**Speaker E**

**Speaker A**

**Speaker B**

