applied sciences

MDPI

Article

Unsupervised Pre-Training for Voice Activation

Aliaksei Kolesau and Dmitrij Šešok *

Department of Information Technologies, Vilnius Gediminas Technical University, Sauletekio al. 11, LT-10223 Vilnius, Lithuania; aliaksei.kolesau@vgtu.lt

* Correspondence: dmitrij.sesok@vgtu.lt

Received: 10 November 2020; Accepted: 30 November 2020; Published: 3 December 2020

Featured Application: The proposed way to use unsupervised pre-training in voice activation could be beneficial in cases of limited data resources, e.g., in low-resource domains or for customizing a product for the end user using his or her voice data. Furthermore, the presented dataset for the Lithuanian language can be used for the further research of voice-related problems in low-resource languages.

Abstract: The problem of voice activation is to find a pre-defined word in the audio stream. Solutions such as keyword spotter "Ok, Google" for Android devices or keyword spotter "Alexa" for Amazon devices use tens of thousands to millions of keyword examples in training. In this paper, we explore the possibility of using pre-trained audio features to build voice activation with a small number of keyword examples. The contribution of this article consists of two parts. First, we investigate the dependence of the quality of the voice activation system on the number of examples in training for English and Russian and show that the use of pre-trained audio features, such as wav2vec, increases the accuracy of the system by up to 10% if only seven examples are available for each keyword during training. At the same time, the benefits of such features become less and disappear as the dataset size increases. Secondly, we prepare and provide for general use a dataset for training and testing voice activation for the Lithuanian language. We also provide training results on this dataset.

Keywords: voice activation; unsupervised learning; pre-training; keyword spotter; neural network; ResNet

1. Introduction

Voice activation systems solve the task of finding predefined keywords or keyphrases in an audio stream [1]. This task has attracted both researchers and industry for decades. Since the task of formulating an algorithm for determining whether a keyphrase has been uttered in an audio stream is difficult to formulate, it is not surprising that heuristic algorithms and machine learning methods have long been used for the voice activation problem.

The history of voice activation models has gone through several important stages in parallel with solving a more general problem of automatic speech recognition (ASR). We would like to highlight the following important moments: the beginning of the use of hidden Markov models back in 1989 [2], the use of neural networks since 1990 [3–5], the use of pattern matching approaches, in particular dynamic time wrapping (DTW) [6], building systems of voice activation for non-English languages such as Chinese [7], Japanese [8], and Iranian [9], publications describing voice activation systems in mass products [10–13], as well as publishing open datasets to compare different approaches [14].

Voice activation systems find applications in various areas: telephony [15], speech spoofing detection [16,17] crime analysis [18], the assistance systems in emergency situations [19], automated management of airports [20], and, naturally, personal voice assistants, built-in in mobile phones and home devices [11].

One of the problems of current state-of-the-art solutions is the use of very large datasets for training neural network, e.g., the authors of [10] used hundreds of thousands samples per keyword, and the

authors of [21] used millions. This presents a question of how to build a high-quality voice activation system in cases when limited training data are available. This can be useful for the following reasons:

- customizing a product by using user-defined keywords, e.g., for personal voice assistants,
- creating voice activation for low-resource languages such as Lithuanian, Latvian, and others.

In this work, we propose to use unsupervised pre-training for building voice activation systems with limited training data. We use the wav2vec method [22] and show that it can improve the quality of the resulting system if there are less than 20 samples per keyword for several datasets, namely Google Speech Commands [14] and our private Russian dataset, even though the wav2vec model was trained only on English recordings. We verify this statement on a new Lithuanian dataset [23], which we collected and present in this work.

2. Related Works

2.1. Low-Resource Keyword Spotting

Keyword spotting in a low-resource setting is a difficult task, which attracts many researches. For example, Reference [24] investigated feature choice for DTW. This research was done to support United Nations humanitarian relief efforts in parts of Africa with severely under-resourced languages. The authors compared multilingual bottleneck features of the model, trained on well-resourced, but out-of-domain languages, and a correspondence autoencoder trained in a zero-resource fashion, as well as their combination. They found that this combination improves the quality of the voice activation system compared to the Mel-frequency cepstral coefficients (MFCC), which are widely used in ASR and voice activation [1].

In [25], the authors applied DTW on Gaussian posteriorgrams from a Gaussian mixture model trained in an unsupervised fashion. The authors of [26] proposed to use tandem acoustic models on different languages to obtain good bottleneck features.

2.2. Unsupervised Pre-Training for Speech

Unsupervised pre-training is one of the methods to cope with limited resources and generally improve the quality of the resulting neural network [27]. The idea is to use a large corpus with no pre-existing labels to learn patterns in data and then to fine-tune the model on the data with labels.

There are works on how to apply pre-training in voice-related problems. For example, the authors of [28] used per-layer pre-training to improve the quality of deep neural network-based ASR.

A promising way to perform unsupervised pre-training for speech is to learn audio features instead of using classical MFCCs or log-Mel filter banks features. For example, a problem-agnostic speech encoder [29] is a feature extractor trained by jointly optimizing multiple self-supervised objectives. Autoregressive predictive coding [30] and contrastive predictive coding [31] are feature extractors trained with objective of predicting some future frames or information about them by having access to the information about the current and some past frames. The authors of these works tested audio features on problems of speech recognition, speaker identification, phone classification and speech translation.

In our paper, we use the wav2vec model [22]. It is a simple multi-layer convolutional neural network optimized via a noise contrastive binary classification task. The authors of [22] reported outperforming the best reported character-based system in the literature while using two orders of magnitude less labeled training data on the ASR task.

To the best of our knowledge, our work is the first attempt to use pre-trained audio features like wav2vec for a voice activation problem in a low-resource setup.

3. Results and Discussion

3.1. Datasets

We used the following three datasets in our experiments:

- English dataset—Google Speech Commands [14],
- Russian dataset—private dataset,
- Lithuanian dataset—collected by us [23].

The Google Speech Commands dataset [14] was released in August 2017 under a Creative Commons license. The dataset contains around 100,000 one second long utterances of 30 short words by thousands of different people, as well as background noise samples such as pink noise, white noise, and human-made sounds. Following the Google implementation [14], our task is to discriminate among 12 classes: "yes", "no", "up", "down", "left", "right", "on", "off", "stop", "go", unknown, and silence.

The Russian dataset is a private dataset that contains around 400,000 one second long utterances of 80 words by 100 different people. These utterances were recorded on mobile devices. The dataset lacks background noise samples, so we reused the samples from the Google Speech Commands dataset [14]. We discriminate the following 12 classes: "один" (one), "два" (two), "три" (three), "четыре" (four), "пять" (five), "да" (yes), "нет" (no), "спасибо" (thanks), "стоп" (stop), "включи" (turn on), unknown, and silence.

Furthermore, specifically for this and future works on voice activation in Lithuanian, we collected a similar dataset for Lithuanian [23]. The collection methodology is described in Section 4. This dataset consists of the recordings of 28 people. Each of them uttered 20 words on a mobile phone. These recordings were segmented into one second long files. The segments between words were used as background noise samples. They contained silence, human-made sound, and background audio such as street or car noises. We chose the following 15 classes: "ne" (no), "ačiū" (thanks), "stop" (stop), "įjunk" (turn on), "išjunk" (turn off), "į viršų" (top), "į apačią" (bottom), "į dešinę" (right), "į kairę" (left), "startas" (start), "pauzė" (pause), "labas" (hello), "iki" (bye), unknown, and silence.

3.2. Model

We used two types of audio features and two types of neural network architectures in our experiments. We used either log-Mel filter banks or pre-trained audio features from the wav2vec model [22].

The log-Mel filter banks features are often chosen for building voice activation or speech recognition systems [1,32]. We used the `kaldi`[33] implementation of feature computation with the following parameters: frame width—25 ms, frame shift—10 ms, number of bins—80. Thus, we got a 98×80 feature matrix by computing log-Mel filter banks on one second samples from the datasets. The method `torchaudio.compliance.kaldi.fbank` can be used in PyTorch [34] to reproduce this computation.

In the case of wav2vec audio features, we uses the pre-trained model from https://github.com/pytorch/fairseq/blob/master/examples/wav2vec/README.md#wav2vec. We got a 98×512 feature matrix as an input for the neural network.

We used the following neural network architectures: a three-layer fully-connected neural network and residual neural networks (ResNets) as described in [32].

Our fully-connected neural network consisted of the following blocks:

- fully-connected layer of size 128,
- rectified linear unit (ReLU) as an activation function [35],
- fully-connected layer of size 64,
- ReLU,

- flattening of a $T \times 64$ matrix in a $64T$ vector, where T is the number of frames in a sample (98 in all our experiments),
- fully-connected layer of size C, where C is the number of classes to discriminate,
- softmax layer.

This neural network architecture is presented on Figure 1.

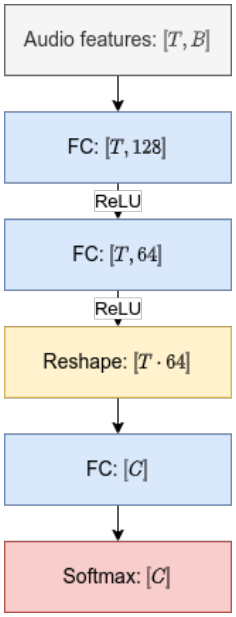


Figure 1. The scheme of the feedforward neural network used in our experiments. The shape of the resulting tensor is specified after each block in square brackets. FC stands for the fully-connected layer. T is the number of frames in a sample (98 in all our experiments). B is the number of channels in the audio features (80 for log-Mel filter banks, 512 for wav2vec features). C is the number of classes to discriminate.

The ResNets that we used in our experiments were based on [36] and repeat the solutions found in [32]. The authors of [36] proposed that it may be easier to learn the residual $H(x) = F(x) + x$ instead of the true mapping $F(x)$, since it is empirically difficult to learn the identity mapping for F when the model has an unnecessary depth. In ResNets, residuals are expressed via connections between layers (see Figure 2), where the input of some layer is added to the output of some downstream layer.

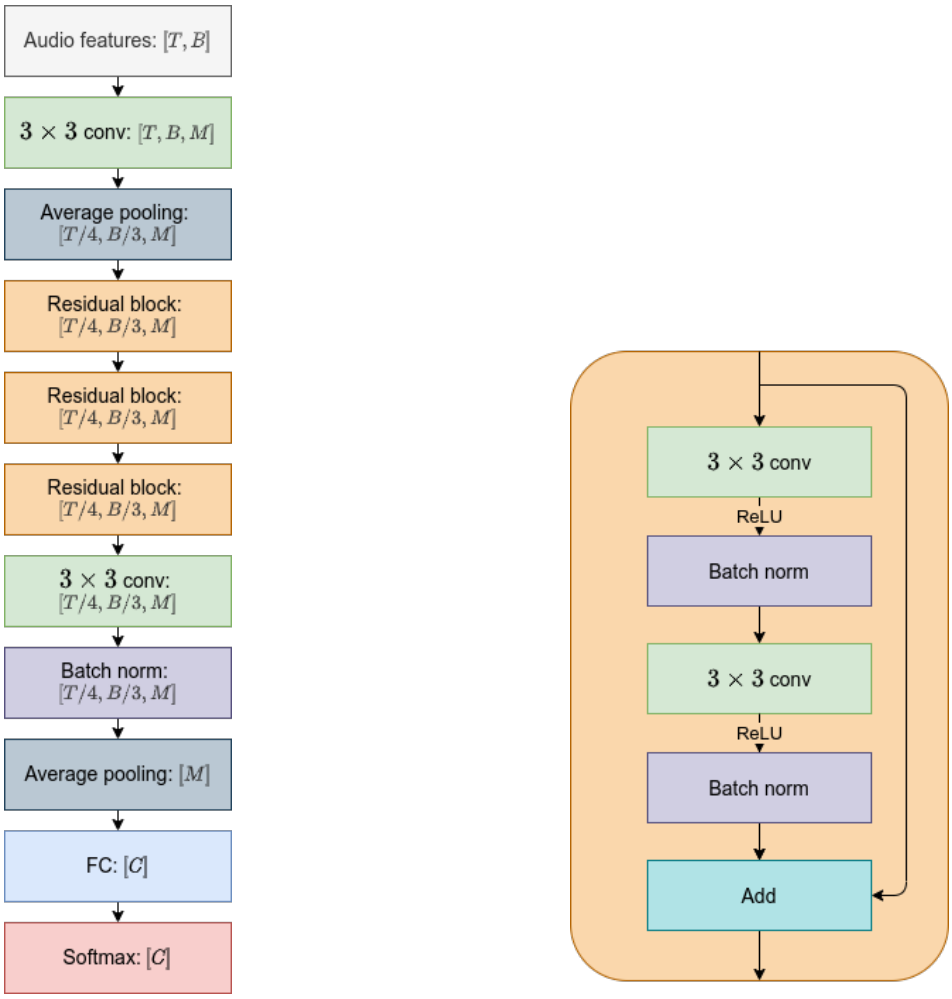
The architectures that we used from [32] consisted of the following blocks:

- bias-free 3×3 convolutional layer,
- optional average pooling layer (e.g., 4×3 layer in `res8`),
- several residual blocks consisting of repeated 3×3 convolutions, ReLUs, and batch normalization layers [37] (see Figure 2b),
- 3×3 convolutional layer,
- batch normalization layer,
- average pooling layer
- fully-connected layer of size C, where C is the number of classes to discriminate,
- softmax-layer.

All the layers were zero-padded. For some variants, dilated convolutions were applied to increase the receptive field of the model. The parameters of all used ResNet architectures can be seen in Table 1.

Table 1. Parameters of the ResNet architectures used in the experiments. The names follow [32].

Architecture Name	Residual Blocks	Feature Maps	Average Pooling	Dilation
`res8`	3	45	(4×3)	no
`res8-narrow`	3	19	(4×3)	no
`res15`	6	45	no	yes
`res15-narrow`	6	19	no	yes
`res26`	12	45	(2×2)	no
`res26-narrow`	12	19	(2×2)	no



(**a**) `res8` (**b**) ResNet block

Figure 2. (**a**) The scheme of the `res8` neural network used in our experiments. The shape of the resulting tensor is specified after each block in square brackets. FC stands for the fully-connected layer. `conv` stands for the convolutional layer. T is the number of frames in a sample. B is the number of channels in the audio features. M is the number of feature maps (45 for `res8`). C is the number of classes to discriminate. Note that the first average pooling layer operates in the time and frequency dimensions, making them smaller, and the second such layer reduces them to one, essentially removing these dimensions. (**b**) shows the detailed schema of each residual block.

The number of trainable parameters in the architectures used are reported in Table 2. More details about the residual architectures can be found in [32].

Table 2. Number of parameters in the architectures used in the experiments for the case of the Google Speech Commands dataset [14] and the log-Mel filter banks. The names of the residual networks follow [32]. FC stands for the fully-connected layer. conv stands for the convolutional layer.

	res8-Narrow	res8	res15-Narrow	res15	res26-Narrow	res26	ff
conv/first FC	171	405	171	405	171	405	10K
residual blocks	19.5K	109K	39K	219K	78K	437K	-
conv/second FC	-	-	3K	18.2K	-	-	8K
softmax	228	540	228	540	228	540	75K
Total	19.K	110K	42.6K	238K	78.4K	438K	93K

3.3. Training Procedure

Our experiments followed exactly the same procedure as the TensorFlow reference for the Google Speech Commands dataset [14]. The Speech Commands Dataset was split into training, validation, and test sets, with 80% training, 10% validation, and 10% test. This resulted in roughly 80,000 examples for training and 10,000 each for validation and testing. For the Russian dataset, these numbers were roughly 320,000 and 40,000. For the Lithuanian dataset [23], we had 326 records for training, 75 for validation, and 88 for testing (we skewed the distribution to ensure more stable test results). For consistency across runs, the SHA1-hashed name of the audio file from the dataset determined the split.

To generate the training data, we followed the Google preprocessing procedure by adding background noise to each sample with a probability of 0.7 at every epoch, where the noise was chosen randomly from the background noises provided in the dataset.

Accuracy was our main metric of quality, which is simply measured as the fraction of classification decisions that are correct. For each input utterance, the model outputs the most likely predicted class.

We ran an extensive random hyperparameter search [38] for all experiments in order to reliably compare audio features and architectures. We used stochastic gradient descent with initial learning rate L, momentum 0.9, and mini-batch size BS (see Appendix A for the specific values of the hyperparameters). The validation metrics (cross-entropy loss and accuracy) were computed every S steps of optimization. The minimal validation accuracy was stored. If the new validation accuracy was bigger than the minimal or if the cross-entropy loss obtained a "not a number" value, the weights of the best (by validation accuracy) step were loaded, but the learning rate dropped by a factor of L'. The training process stopped when the learning rate drop happened the sixth time. The test accuracy was computed exactly once: on the best model by the validation accuracy at the end of the training process. We report the test accuracy in this work.

We chose $-\log_{10} L$ from $U\{0,3\}$, $\log_2 \mathrm{BS}$ from $U\{4,7\}$, $\log_2 S$ from $U\{3,12\}$, and L' from $U\{1.1,10.0\}$, where U is a uniform distribution (discrete uniform distribution in the case of S and BS).

The code is available at https://github.com/kolesov93/keyword_spotter_train.

3.4. Results

In this section, we present only the test metrics in order to not clutter the description. For the hyperparameters' choice, see Appendix A.

In order to get baseline metrics, we ran experiments on full datasets with both log-Mel filter banks and wav2vec features. The best results of these runs are presented in Table 3 for the English dataset. We got slightly better results than in [32]. This can be explained by the following reasons:

- The Google Speech Commands dataset [14] was extended since its publication,
- we used a more extensive hyperparameter search.

We made the following conclusions from the results:

- wav2vec audio features give a competitive result for the voice activation problem with very simple downstream models such as the feedforward neural network,

- the profit of unsupervised pre-training vanishes as the model gets more sophisticated and deep.

We repeated the same experiments for the Russian dataset and got similar results (Table 4): `ff` and `ResNet8-narrow` as the simplest models got better results with wav2vec audio features. However, the overall best result was still with log-Mel filter banks: 97.22%. The best result of wav2vec runs was 96.62%, which was worse, but still very competitive.

It is worth noting that wav2vec model was trained on the Librespeech dataset [39], which contains only English audio books. It is promising that using this model, it was possible to get good accuracy both on the Russian and Lithuanian datasets (see Table 5). Moreover, we got better results on the Lithuanian dataset using wav2vec than using log-Mel filter banks (90.77% vs. 89.23%).

Table 3. Test accuracy on the full English dataset.

Architecture	Log-Mel Filter Banks	wav2vec
`ff`	73.14%	**97.20%**
`res8-narrow`	92.61%	**95.91%**
`res8`	95.07%	**95.79%**
`res15-narrow`	**96.44%**	95.70%
`res15`	**97.03%**	96.07%
`res26-narrow`	**96.75%**	91.45%
`res26`	**97.46%**	95.97%

Table 4. Test accuracy on the full Russian dataset.

Architecture	Log-Mel Filter Banks	wav2vec
`ff`	89.15%	**94.87%**
`res8-narrow`	94.70%	**95.81%**
`res8`	**96.72%**	96.26%
`res15-narrow`	**94.53%**	94.24%
`res15`	**97.22%**	96.03%
`res26-narrow`	95.42%	**96.62%**
`res26`	**95.81%**	95.03%

Table 5. Test accuracy on the Lithuanian dataset.

Architecture	Log-Mel Filter Banks	wav2vec
`ff`	72.31%	**78.46%**
`res8-narrow`	73.85%	**84.62%**
`res8`	78.46%	**90.77%**
`res15-narrow`	**83.08%**	80.00%
`res15`	**89.23%**	86.15%
`res26-narrow`	**83.08%**	72.31%
`res26`	80.00%	**83.08%**

Next, we ran experiments with a small amount of training data. In order to do that, we limited the number of training samples per keyword by 3, 5, 7, 10, and 20 for all the datasets. Note that the limit of 20 is effectively the same as using the whole dataset for the Lithuanian language. The size of the validation and test sets remained the same in order to get reliable and comparable results. We used random search with all the models and report the test accuracy of the best runs. The motivation of these experiments goes as follows. First of all, the authors of [22] reported that they got state-of-the-art results in automatic speech recognition with unsupervised pre-training in the case when limited training data were available. Secondly, our first set of experiments showed that wav2vec audio features are superior when the machine learning model is simple. Simpler models tend to perform better when a dataset is small. Therefore, it might be profitable to use unsupervised pre-trained audio features in this scenario. The results of these experiments are summarized in Table 6.

Table 6. Test accuracy on limited datasets for English (en), Russian (ru) and Lithuanian (lt) datasets. The best model by test accuracy is specified in brackets. Note that the limit of 20 is effectively the same as using the whole dataset for the Lithuanian language.

Limit	Dataset	Log-Mel Filter Banks	wav2vec	Relative Improvement
3	en	39.30% (`res15`)	37.05% (`res8`)	−5%
3	ru	31.72% (`res15-narrow`)	48.51% (`res15`)	53%
3	lt	44.62% (`res15`)	56.92% (`ff`)	27%
5	en	45.91% (`res15`)	59.41% (`res15-narrow`)	29%
5	ru	40.42% (`res8-narrow`)	57.65% (`ff`)	42%
5	lt	55.38% (`res15-narrow`)	67.69% (`ff`)	22%
7	en	50.55% (`res15`)	60.41% (`res8-narrow`)	20%
7	ru	57.69% (`res26`)	63.88% (`res15-narrow`)	11%
7	lt	58.46% (`res8`)	67.69% (`res15`)	15%
10	en	54.40% (`res15`)	61.05% (`res8`)	12%
10	ru	54.60% (`ff`)	58.68% (`res26`)	7%
10	lt	72.31% (`res15`)	78.46% (`res15`)	9%
20	en	74.22% (`res15`)	75.63% (`res8-narrow`)	2%
20	ru	67.43% (`res15`)	73.79% (`res26`)	9%
20	lt	89.23% (`res15`)	90.77% (`res8`)	1%

It can be seen that the use of pre-trained audio features as wav2vec increases the system accuracy by approximately 10% when up to 10 samples are used per keyword both for the English and Russian language despite the fact that the model was only trained on English audio records. The increase is even bigger if five samples are used. It almost vanishes if up to 20 samples are used.

4. Collecting the Lithuanian Dataset

In order to boost voice activation research in Lithuanian, we prepared a dataset in the format of Google Speech Commands [14]. This section describes how we carried out the data collection and preparation.

Firstly, we chose 20 keywords to record by translating some of the Google Speech Commands [14]: “nulis” (zero), “vienas” (one), “du” (two), “trys” (three), “keturi” (four), “penki” (five), “taip” (yes), **“ne” (no)**, **“ačiū” (thanks)**, **“stop” (stop)**, **“įjunk” (turn on)**, **“išjunk” (turn off)**, **“į viršų” (top)**, **“į apačią” (bottom)**, **“į dešinę” (right)**, **“į kairę” (left)**, **“startas” (start)**, **“pauzė” (pause)**, **“labas” (hello)**, **“iki” (bye)**. Words in bold were selected as target classes for the voice activation problem. We chose these words to discriminate in order to ensure a challenging problem of differentiating words with similar word parts: compare “įjunk” (turn on) and “išjunk”; the starts of keyphrases “į viršų, “į apačią”, “į dešinę”, “į kairę”; the ends of “startas” and “labas”. Very short keywords “iki” and “ne” also increase the complexity of the task.

We asked several volunteers to record these words in the specified order on their mobile devices. We did not restrict the speed of pronunciation, but asked to make pauses between words. We collected 28 records ranging from 24 to 64 s. We checked all the records for the correct order of words.

The next step was to segment these records into one second samples. We used Audacity v2.2.1 (Audacity ® software is ©1999–2020 Audacity Team. The name Audacity® is a registered trademark of Dominic Mazzoni) in the following way:

- apply the “Sound Finder” analysis tool with default parameters (Figure 3),
- if 20 sound segments were not found, remove all segments, and manually reselect them,
- listen to each sound segment; move the start or the end of the segment if necessary,
- make sure that segments have numbers from 1 to 20 as labels,
- export labels to a separate text file.

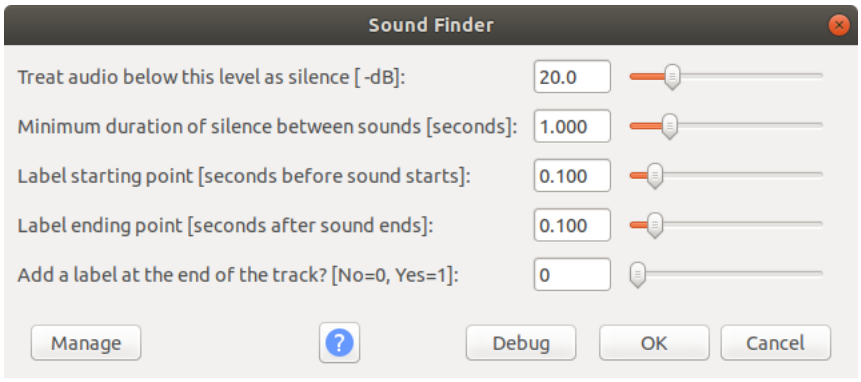


Figure 3. Parameters of the "Sound Finder" analysis tool.

The resulting text file has the following format: "s_i e_i `label`", where s_i is the start of the i-th segment in seconds and e_i is the end. Using these files, we prepared the dataset with a following algorithm:

- skip the current segment if $e_i - s_i > 1$, because the word itself is longer than one second,
- skip the current segment if $s_{i+1} - e_{i-1} < 1$, because such a one second interval will contain parts of neighboring words,
- for each segment, compute the range $[A_i, B_i]$ where the start of the one second interval can be chosen ($\varepsilon = 0.1$ in order to make at least a short pause before the start of the word, $e_0 = 0$; s_{i+1} for the last segment is equal to the whole utterance duration):

$$A_i = \max(e_{i-1}, \min(s_{i+1} - 1 - \varepsilon, s_i - \varepsilon)), \tag{1}$$

$$B_i = s_i - \varepsilon. \tag{2}$$

- pick S_i uniformly from $[A_i, B_i]$, and cut the segment $[S_i, S_i + 1]$ as a separate sample for the i-th word.

For each audio segment between the words with a duration bigger than one second, we uniformly picked exactly one one second sub-segment and used it as a background noise. We got 292 no-speech segments in such a fashion.

The raw recordings, text files with labels, code to perform the cutting, and the dataset itself are available at https://github.com/kolesov93/lt_speech_commands.

5. Conclusions

In this work, we proposed to use pre-trained audio features for the voice activation systems in the case of limited training data. The experiments on the Google Speech Commands dataset [14] showed that the proposed audio features improve the accuracy of the voice activation system by 10% when the number of samples per keyword is seven or less and by 29% if the number of samples per keyword is five or less both for the English and Russian datasets. It is also worth noting that we only used the wav2vec model pre-trained on English audio recordings. The improvement however vanished when the whole datasets were used, which may indicate the limits of the proposed method. Furthermore, we collected a Lithuanian dataset [23] for voice activation and reproduced our results on it.

In future works, other methods of unsupervised pre-training for voice activation can be investigated and compared to the wav2vec method. Additionally, the influence of domain mismatch between unsupervised pre-training of audio features and the downstream voice activation task can be studied.

Author Contributions: Conceptualization, methodology, software, writing, original draft preparation, visualization: A.K.; writing, review and editing, supervision, project administration, data curation: D.Š. All authors read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

In this Appendix, we provide the choice of the hyperparameters for the results presented in Section 3.4. See Table A1 for the results on the Google Speech Commands dataset [14], Table A2 for the results on the Russian dataset, and Table A3 for the results on the Lithuanian dataset [23].

Table A1. Test metrics and hyperparameters on the Google Speech Commands dataset [14].

Limit	Features	BS	S	Architecture	L	L'	Accuracy	CE Loss
no	fbank	64	1024	`ff`	0.0018	4.0535	73.14%	0.936
no	wav2vec	32	1024	`res26-narrow`	0.0352	7.6938	91.45%	0.279
no	fbank	64	1024	`res8-narrow`	0.0161	7.3268	92.61%	0.242
no	fbank	16	2048	`res8`	0.0067	2.1576	95.07%	0.159
no	wav2vec	16	2048	`res15-narrow`	0.0198	6.6472	95.70%	0.143
no	wav2vec	32	2048	`res8`	0.0036	9.6358	95.79%	0.131
no	wav2vec	16	2048	`res8-narrow`	0.0164	1.2408	95.91%	0.135
no	wav2vec	64	512	`res26`	0.0042	5.3919	95.97%	0.130
no	wav2vec	64	1024	`res15`	0.0983	6.4306	96.07%	0.113
no	fbank	32	2048	`res15-narrow`	0.0128	7.9980	96.44%	0.105
no	fbank	32	2048	`res26-narrow`	0.1515	5.7268	96.75%	0.099
no	fbank	64	512	`res15`	0.0035	1.4772	97.03%	0.105
no	wav2vec	16	2048	`ff`	0.0056	2.8667	97.20%	0.087
no	fbank	64	2048	`res26`	0.0578	4.2733	97.46%	0.076
3	wav2vec	16	512	`res8`	0.0015	6.8410	37.05%	1.854
3	fbank	64	32	`res15`	0.0419	5.5069	39.30%	1.999
5	fbank	16	256	`res15`	0.2014	7.6873	45.91%	1.897
7	fbank	16	256	`res15`	0.3133	4.3166	50.55%	2.120
10	fbank	64	128	`res15`	0.0777	6.4338	54.40%	1.637
5	wav2vec	16	1024	`res15-narrow`	0.0050	7.6948	59.41%	1.530
7	wav2vec	64	512	`res8-narrow`	0.0300	4.5449	60.41%	1.378
10	wav2vec	16	2048	`res8`	0.0195	9.1938	61.05%	2.010
20	fbank	16	2048	`res15`	0.0237	3.3050	74.22%	0.975
20	wav2vec	32	1024	`res8-narrow`	0.0099	4.4134	75.63%	0.980

Table A2. Test metrics and hyperparameters on the Russian dataset.

Limit	Features	BS	S	Architecture	L	L'	Accuracy	CE Loss
3	fbank	16	32	`res15-narrow`	0.1635	4.1339	31.72%	2.330
5	fbank	16	256	`res8-narrow`	0.0312	4.9367	40.42%	1.932
3	wav2vec	16	256	`res15`	0.0179	1.8520	48.51%	2.490
10	fbank	32	16	`ff`	0.1637	5.8915	54.60%	1.569
5	wav2vec	16	128	`ff`	0.0067	6.6971	57.65%	1.970
7	fbank	16	512	`res26`	0.0060	7.8152	57.69%	1.463
10	wav2vec	32	512	`res26`	0.3383	2.8521	58.68%	1.873
7	wav2vec	16	1024	`res15-narrow`	0.0046	2.5055	63.88%	1.148
20	fbank	16	2048	`res15`	0.0012	1.3961	67.43%	1.067
20	wav2vec	64	2048	`res26`	0.0122	4.5916	73.79%	0.983

Table A3. Test metrics and hyperparameters on the Lithuanian dataset [23].

Limit	Features	BS	S	Architecture	L	L'	Accuracy	CE Loss
no	fbank	32	512	`ff`	0.0017	5.1070	72.31%	1.241
no	wav2vec	16	256	`ff`	0.0021	1.4282	78.46%	0.746
no	fbank	32	512	`res15`	0.1049	4.2465	89.23%	0.448
no	wav2vec	16	512	`res15`	0.0092	3.4831	86.15%	0.317
no	fbank	64	512	`res15-narrow`	0.0744	5.2013	83.08%	0.504
no	wav2vec	64	2048	`res15-narrow`	0.0049	3.2217	80.00%	0.629
no	fbank	32	1024	`res26`	0.0657	2.0357	80.00%	0.680
no	wav2vec	16	2048	`res26`	0.0055	6.8291	83.08%	0.408
no	fbank	32	1024	`res26-narrow`	0.0695	6.2596	83.08%	0.492
no	wav2vec	16	256	`res26-narrow`	0.0299	3.0330	72.31%	0.726
no	fbank	16	2048	`res8`	0.0114	2.9399	78.46%	0.610
no	wav2vec	32	512	`res8`	0.1233	9.8922	90.77%	0.130
no	fbank	16	2048	`res8-narrow`	0.1734	8.3757	73.85%	0.752
no	wav2vec	64	2048	`res8-narrow`	0.0812	7.8581	84.62%	0.504
3	fbank	16	256	`res15`	0.0055	5.7572	44.62%	1.807
3	wav2vec	64	16	`ff`	0.0019	3.8993	56.92%	1.070
5	fbank	32	512	`res15-narrow`	0.0396	9.6257	55.38%	1.192
5	wav2vec	16	32	`ff`	0.0064	6.2748	67.69%	1.137
7	fbank	16	2048	`res8`	0.1429	5.2105	58.46%	1.089
7	wav2vec	16	512	`res15`	0.0135	7.4800	67.69%	0.709
10	fbank	32	64	`res15`	0.0498	8.0316	72.31%	0.659
10	wav2vec	64	2048	`res15`	0.0626	6.1930	78.46%	0.866

References

1. Kolesau, A.; Šešok, D. Voice Activation Systems for Embedded Devices: Systematic Literature Review. *Informatica* **2020**, 65–88. [CrossRef]
2. Rohlicek, J.R.; Russell, W.; Roukos, S.; Gish, H. Continuous hidden Markov modeling for speaker-independent word spotting. *Int. Conf. Acoust. Speech Signal Process.* **1989**, *1*, 627–630. [CrossRef]
3. Morgan, D.P.; Scofield, C.L.; Lorenzo, T.M.; Real, E.C.; Loconto, D.P. A keyword spotter which incorporates neural networks for secondary processing. *Int. Conf. Acoust. Speech Signal Process.* **1990**, *1*, 113–116. [CrossRef]
4. Morgan, D.P.; Scofield, C.L.; Adcock, J.E. Multiple neural network topologies applied to keyword spotting. In Proceedings of the ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing, Toronto, ON, Canada, 14–17 April 1991; pp. 313–316. [CrossRef]
5. Naylor, J.A.; Huang, W.Y.; Nguyen, M.; Li, K.P. The application of neural networks to wordspotting. In Proceedings of the Conference Record of the Twenty-Sixth Asilomar Conference on Signals, Systems Computers, Pacific Grove, CA, USA, 26–28 October 1992; pp. 1081–1085. [CrossRef]
6. Zeppenfeld, T.; Waibel, A.H. A hybrid neural network, dynamic programming word spotter. In Proceedings of the ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, San Francisco, CA, USA 23–26 March 1992; pp. 77–80. [CrossRef]
7. Zheng, F.; Xu, M.; Mou, X.; Wu, J.; Wu, W.; Fang, D. HarkMan—A vocabulary-independent keyword spotter for spontaneous Chinese speech. *J. Comput. Sci. Technol.* **1999**, *14*, 18–26. [CrossRef]
8. Ida, M.; Yamasaki, R. An evaluation of keyword spotting performance utilizing false alarm rejection based on prosodic information. In Proceedings of the 5th International Conference on Spoken Language Processing, Incorporating the 7th Australian International Speech Science and Technology Conference, Sydney Convention Centre, Sydney, Australia, 30 November–4 December 1998.
9. Shokri, A.; Tabibian, S.; Akbari, A.; Nasersharif, B.; Kabudian, J. A robust keyword spotting system for Persian conversational telephone speech using feature and score normalization and ARMA filter. In Proceedings of the 2011 IEEE GCC Conference and Exhibition (GCC), Dubai, UAE, 19–22 February 2011; pp. 497–500. [CrossRef]

10. Chen, G.; Parada, C.; Heigold, G. Small-footprint keyword spotting using deep neural networks. In Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, 4–9 May 2014; pp. 4087–4091.
11. Gruenstein, A.; Alvarez, R.; Thornton, C.; Ghodrat, M. A Cascade Architecture for Keyword Spotting on Mobile Devices. *arXiv* **2017**, arXiv:1712.03603.
12. Guo, J.; Kumatani, K.; Sun, M.; Wu, M.; Raju, A.; Strom, N.; Mandal, A. Time-Delayed Bottleneck Highway Networks Using a DFT Feature for Keyword Spotting. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, 15–20 April 2018; pp. 5489–5493. [CrossRef]
13. Wu, M.; Panchapagesan, S.; Sun, M.; Gu, J.; Thomas, R.; Vitaladevuni, S.N.P.; Hoffmeister, B.; Mandal, A. Monophone-Based Background Modeling for Two-Stage On-Device Wake Word Detection. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, 15–20 April 2018; pp. 5494–5498. [CrossRef]
14. Warden, P. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv* **2018**, arXiv:1804.03209.
15. Shokri, A.; Davarpour, M.H.; Akbari, A.; Nasersharif, B. Detecting keywords in Persian conversational telephony speech using a discriminative English keyword spotter. In Proceedings of the IEEE International Symposium on Signal Processing and Information Technology, Athens, Greece, 12–15 December 2013; IEEE Computer Society: Washington, DC, USA, 2013; pp. 272–276. [CrossRef]
16. Wang, X.; Yamagishi, J.; Todisco, M.; Delgado, H.; Nautsch, A.; Evans, N.; Sahidullah, M.; Vestman, V.; Kinnunen, T.; Lee, K.; et al. ASVspoof 2019: A large-scale public database of synthesized, converted and replayed speech. *Comput. Speech Lang.* **2020**, *64*. [CrossRef]
17. Gomez-Alanis, A.; Gonzalez-Lopez, J.A.; Peinado, A.M. A Kernel Density Estimation Based Loss Function and its Application to ASV-Spoofing Detection. *IEEE Access* **2020**, *8*, 108530–108543. [CrossRef]
18. Kavya, H.P.; Karjigi, V. Sensitive keyword spotting for crime analysis. In Proceedings of the 2014 IEEE National Conference on Communication, Signal Processing and Networking (NCCSN), Palakkad, India, 10–12 October 2014; pp. 1–6. [CrossRef]
19. Zhu, C.; Kong, Q.; Zhou, L.; Xiong, G.; Zhu, F. Sensitive keyword spotting for voice alarm systems. In Proceedings of the 2013 IEEE International Conference on Service Operations and Logistics, and Informatics, Dongguan, China, 28–30 July 2013; pp. 350–353. [CrossRef]
20. Tabibian, S. A voice command detection system for aerospace applications. *I. J. Speech Technol.* **2017**, *20*, 1049–1061. [CrossRef]
21. Jose, C.; Mishchenko, Y.; Sénéchal, T.; Shah, A.; Escott, A.; Vitaladevuni, S.N.P. Accurate Detection of Wake Word Start and End Using a CNN. *Proc. Interspeech* **2020**, *2020*, 3346–3350. [CrossRef]
22. Schneider, S.; Baevski, A.; Collobert, R.; Auli, M. wav2vec: Unsupervised Pre-Training for Speech Recognition. *Proc. Interspeech* **2019**, *2019*, 3465–3469. [CrossRef]
23. Kolesau, A.; Šešok, D. Lithuanian Speech Commands Dataset. Available online: https://github.com/kolesov93/lt_speech_commands (accessed on 11 November 2020).
24. Menon, R.; Kamper, H.; van der Westhuizen, E.; Quinn, J.; Niesler, T. Feature Exploration for Almost Zero-Resource ASR-Free Keyword Spotting Using a Multilingual Bottleneck Extractor and Correspondence Autoencoders. *Proc. Interspeech* **2019**, *2019*, 3475–3479, 10.21437/Interspeech.2019-1665.
25. Zhang, Y.; Glass, J.R. Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams. In Proceedings of the 2009 IEEE Workshop on Automatic Speech Recognition Understanding, Merano, Italy, 13 November–17 December 2009; pp. 398–403.
26. Knill, K.; Gales, M.; Ragni, A.; Rath, S. Language independent and unsupervised acoustic models for speech recognition and keyword spotting. In Proceedings of the Annual Conference of the International Speech Communication Association, Singapore, 14–18 September 2014; pp. 16–20.
27. Erhan, D.; Bengio, Y.; Courville, A.; Manzagol, P.A.; Vincent, P.; Bengio, S. Why Does Unsupervised Pre-training Help Deep Learning? *J. Mach. Learn. Res.* **2010**, *11*, 625–660. [CrossRef]
28. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.; Mohamed, A.r.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Kingsbury, B.; et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *IEEE Signal Process. Mag.* **2012**, *29*, 82–97. [CrossRef]

29. Pascual, S.; Ravanelli, M.; Serrà, J.; Bonafonte, A.; Bengio, Y. Learning Problem-Agnostic Speech Representations from Multiple Self-Supervised Tasks. *Proc. Interspeech* **2019**, *2019*, 161–165. [CrossRef]
30. Chung, Y.A.; Glass, J.R. Generative Pre-Training for Speech with Autoregressive Predictive Coding. In Proceedings of the ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 3497–3501.
31. van den Oord, A.; Li, Y.; Vinyals, O. Representation Learning with Contrastive Predictive Coding. *arXiv* **2018**, arXiv:1807.03748.
32. Tang, R.; Lin, J. Deep Residual Learning for Small-Footprint Keyword Spotting. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018; pp. 5484–5488. [CrossRef]
33. Povey, D.; Ghoshal, A.; Boulianne, G.; Burget, L.; Glembek, O.; Goel, N.; Hannemann, M.; Motlicek, P.; Qian, Y.; Schwarz, P.; et al. The Kaldi Speech Recognition Toolkit. In Proceedings of the IEEE 2011 Workshop on Automatic Speech Recognition and Understanding. IEEE Signal Processing Society, Big Island, HI, USA, 11–15 December 2011; IEEE Catalog No.: CFP11SRW-USB.
34. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; GiMelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Curran Associates, Inc.: New York, NY, USA, 2019; pp. 8024–8035.
35. Nair, V.; Hinton, G. Rectified Linear Units Improve Restricted Boltzmann Machines. *ICML* **2010**, *27*, 807–814.
36. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
37. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Conference on International Conference on Machine Learning—Volume 37. JMLR.org, Lille, France, 6–11 July 2015; pp. 448–456.
38. Bergstra, J.; Bengio, Y. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
39. Panayotov, V.; Chen, G.; Povey, D.; Khudanpur, S. Librispeech: An ASR corpus based on public domain audio books. In Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, QLD, Australia, 19–24 April 2015; pp. 5206–5210. [CrossRef]