VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

Aliaksei KOLESAU

IMPROVING THE EFFECTIVENESS OF VOICE ACTIVATION SYSTEMS WITH MACHINE LEARNING METHODS

DOCTORAL DISSERTATION

TECHNOLOGICAL SCIENCES,
INFORMATICS ENGINEERING (T 007)

Vilnius, 2022

Doctoral dissertation was prepared at Vilnius Gediminas Technical University in 2017–2022.

Supervisor

Assoc. Prof. Dr Dmitrij ŠEŠOK (Vilnius Gediminas Technical University, Informatics Engineering – T 007).

The Dissertation Defense Council of Scientific Field of Informatics Engineering of Vilnius Gediminas Technical University:

Chairman

Prof. Dr Arnas KAČENIAUSKAS (Vilnius Gediminas Technical University, Informatics Engineering – T 007).

Members:

Assoc. Prof. Dr Olga KURASOVA (Vilnius University, Informatics Engineering – T 007),

Prof. Dr Rytis MASKELIŪNAS (Kaunas University of Technology, Informatics Engineering – T 007),

Prof. Dr Dalius MAŽEIKA (Vilnius Gediminas Technical University, Informatics Engineering – T 007),

Prof. Dr Audris MOCKUS (University of Tennessee, USA, Informatics – N 009).

The dissertation will be defended at the public meeting of the Dissertation Defense Council of Informatics Engineering in the Senate Hall of Vilnius Gediminas Technical University at **2 p. m. on 30 August 2022**.

Address: Saulėtekio al. 11, LT-10223 Vilnius, Lithuania.
Tel.: +370 5 274 4956; fax +370 5 270 0112; e-mail: doktor@vilniustech.lt

A notification on the intend defending of the dissertation was send on 29 July 2022.
A copy of the doctoral dissertation is available for review at Vilnius Gediminas Technical University repository http://dspace.vilniustech.lt, at the Library of Vilnius Gediminas Technical University (Saulėtekio al. 14, LT-10223 Vilnius, Lithuania) and at the Library of Kaunas University of Technology (K. Donelaičio st. 20, LT-44239 Kaunas, Lithuania).

Vilnius Gediminas Technical University book No 2022-033-M

doi:10.20334/2022-033-M

aliaksei.kolesau@vilniustech.lt

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Aliaksei KOLESAU

BALSO AKTYVAVIMO SISTEMŲ EFEKTYVUMO GERINIMAS NAUDOJANT MAŠININIO MOKYMOSI METODUS

DAKTARO DISERTACIJA

TECHNOLOGIJOS MOKSLAI,
INFORMATIKOS INŽINERIJA (T 007)

Vilnius, 2022

Disertacija rengta 2017–2022 metais Vilniaus Gedimino technikos universitete.

Vadovas

doc. dr. Dmitrij ŠEŠOK (Vilniaus Gedimino technikos universitetas, informatikos inžinerija – T 007).

Vilniaus Gedimino technikos universiteto Informatikos inžinerijos mokslo krypties disertacijos gynimo taryba:

Pirmininkas

prof. dr. Arnas KAČENIAUSKAS (Vilniaus Gedimino technikos universitetas, informatikos inžinerija – T 007).

Nariai:

doc. dr. Olga KURASOVA (Vilniaus universitetas, informatikos inžinerija – T 007),

prof. dr. Rytis MASKELIŪNAS (Kauno technologijos universitetas, informatikos inžinerija – T 007),

prof. dr. Dalius MAŽEIKA (Vilniaus Gedimino technikos universitetas, informatikos inžinerija – T 007),

prof. dr. Audris MOCKUS (Tenesio universitetas, JAV, informatika – N 009).

Disertacija bus ginama viešame Informatikos inžinerijos mokslo krypties disertacijos gynimo tarybos posėdyje **2022 m. rugpjūčio 30 d. 14 val**. Vilniaus Gedimino technikos universiteto senato posėdžių salėje.

Adresas: Saulėtekio al. 11, LT-10223 Vilnius, Lietuva.
Tel.: (8 5) 274 4956; faksas (8 5) 270 0112; el. paštas doktor@vilniustech.lt

Pranešimai apie numatomą ginti disertaciją išsiųsti 2022 m. liepos 29 d.

Disertaciją galima peržiūrėti Vilniaus Gedimino technikos universiteto talpykloje http://dspace.vilniustech.lt, Vilniaus Gedimino technikos universiteto bibliotekoje (Saulėtekio al. 14, LT-10223 Vilnius, Lietuva) ir Kauno technologijos universiteto bibliotekoje (K. Donelaičio g. 20, LT44239 Kaunas, Lietuva).

Abstract

Modern devices are more frequently equipped with voice control. Using speech to operate is a natural and efficient approach. However, incorporating voice control poses the following problem: how users should initiate a dialogue. One way is to process the whole audio stream. This solution raises privacy concerns and requires a lot of computational and network resources. Also, the problem of differentiating the user's command and non-relevant speech appears in such a setup. The alternative is to use a voice activation: initiating a voice dialogue by pronouncing the pre-defined keyword or keyphrase. The problem of finding such pre-defined keyphrase in the audio stream is called "keyword spotting" and can be adequately solved with the computational resources in modern embedded devices. This mitigates the privacy problems, requires less computational resources, and simplifies the problem of detecting a non-relevant speech: if the keyphrase is a-prior rare, its utterance is a good signal, that the user wants to start an interaction.

Since the task of formulating an algorithm for determining whether a keyphrase has been uttered in an audio stream is difficult, it is not surprising that machine learning methods have long been used for voice activation systems. Most of modern research in keyword spotting is focused on well-resourced setups and tuning and proposing deep learning methods to improve the detection quality.

This dissertation consists of an introduction, three main chapters, and general conclusions. The first chapter reviews existing research on voice activation systems, proposes the general structure of such a system and introduces the Lithuanian dataset for training a voice activation system. The second chapter investigates acoustic feature pipelines used in modern voice activation systems, studies the simplification trend in acoustic feature extraction, and shows the importance of pipeline hyperparameter tuning for achieving the best results. Also, the chapter discusses acoustic units used in keyword spotting and how to use training samples with these units effectively. The approach to detect keyword repeats to improve the recall of a voice activation system is proposed at the end of the second chapter. Chapter three presents new methods to build a voice activation system in a low-resource setup.

The performed experiments and analysis have demonstrated that all main parts of a voice activation system (acoustic feature pipeline, acoustic model, and decoding) are important for achieving a good detection quality. Also, the proposed pre-training method for voice activation in a low-resource setup shows the accuracy improvements of a 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. Furthermore, the best accuracy for the Lithuanian dataset was improved from 89.23% to 93.85% by the proposed joint training method.

Reziumė

Vis daugiau modernių įrenginių turi valdymo balsu galimybę. Kalbos naudojimas yra natūralus ir efektyvus prietaiso valdymo būdas. Tačiau realizuojant valdymo balsu funkcionalumą iškyla problema: kaip vartotojai turėtų pradėti dialogą. Vienas iš būdų yra apdoroti visą garso srautą. Šis sprendimas kelia susirūpinimą dėl konfidencialumo bei reikalauja daug skaičiavimo ir tinklo išteklių. Taip pat atsiranda vartotojo komandų ir nesusijusios kalbos išskyrimo problema. Alternatyva yra naudoti balso dialogo inicijavimą ištariant iš anksto nustatytą reikšminį žodį arba reikšminę frazę. Tokios, iš anksto nustatytos, reikšminės frazės radimo garso sraute problema vadinama reikšminių žodžių aptikimu (angl. *keyword spotting*) ir gali būti tinkamai išspręsta naudojant šiuolaikinių įterptųjų įrenginių (angl. *embedded devices*) skaičiavimo išteklius. Tai sumažina konfidencialumo problemas, reikalauja mažiau skaičiavimo resursų ir supaprastina nesusijusios kalbos aptikimo problemą: jei reikšminė frazė yra *a priori* reta, jos ištarimas yra geras signalas, kad vartotojas nori pradėti sąveiką.

Atsižvelgiant į tai, kad užduotis suformuluoti algoritmą, siekiant nustatyti, ar garso sraute buvo ištarta reikšminė frazė, yra sunkiai formalizuojama, nenuostabu, kad euristiniai algoritmai ir mašininio mokymosi metodai plačiai taikomi balso aktyvavimo problemai spręsti. Absoliuti dauguma šiuolaikinių tyrimų šioje srityje naudoja gerai paruoštus resursus.

Disertaciją sudaro įvadas, trys pagrindinės dalys bei išvados. Pirmoje dalyje pateikta balso aktyvavimo sistemų mokslinių tyrimų apžvalga. Šios dalies pabaigoje siūloma bendra tokios sistemos struktūra. Antroje dalyje pateikiamas akustinių požymių, naudojamų šiuolaikinėse balso aktyvavimo sistemose, tyrimas. Šioje dalyje taip pat pristatomas autoriaus sukurtas balso aktyvavimo sistemos duomenų rinkinys, skirtas lietuvių kalbai. Taip pat aptariami akustiniai vienetai, kurie naudojami reikšminiams žodžiams aptikti. Siūlomas reikšminių žodžių pasikartojimų aptikimo būdas, siekiant pagerinti balso aktyvavimo sistemos atsaką. Tai pasiekiama, keičiant balso aktyvavimo sistemos dekodavimo stadiją. Trečioje dalyje pateikiami keli nauji balso aktyvavimo sistemų kūrimo būdai, skirti darbui su mažais ištekliais.

Atlikti eksperimentai ir analizė parodė, kad visos pagrindinės balso aktyvavimo sistemos dalys: akustinių požymių gavimo algoritmas, akustinis modelis, dekodavimas yra svarbios siekiant gauti gerą aptikimo kokybę. Taip pat siūlomas išankstinio paruošimo metodas, naudojant žemus resursus, parodo, kad balso aktyvavimo sistemos tikslumas padidėja 10 %, kai pavyzdžių skaičius reikšminiam žodžiui yra 7 ir mažiau, ir 29 %, kai pavyzdžių skaičius reikšminiam žodžiui yra 5 ir mažiau. Be to, geriausias lietuvių kalbos duomenų rinkinio tikslumas buvo pagerintas nuo 89,23 % iki 93,85 %.

Notations

Abbreviations

AUC – Area Under the Curve;
ASR – Automatic Speech Recognition;
AMA – Autoregressive Moving Average;
CPU – Central Processing Unit;
CMVN – Cepstral Mean and Variance Normalization;
CNN – Convolutional Neural Network;
CE – Cross-Entropy;
DSP – Digital Signal Processor;
DCT – Discrete Cosine Transform;
DFT – Discrete Fourier Transform;
DTW – Dynamic Time Warping;
EER – Equal Error Rate;
FAR – False Alarm Rate;
FPR – False Positive Rate;
FRR – False Reject Rate;
f-MLLR – Feature Space Maximum Likelihood Linear Regression;
FOM – Figure of Merit;
GMM – Gaussian Mixture Model;
GSC – Google Speech Commands dataset;

HMM – Hidden Markov Model;
LDA – Linear Discriminant Analysis;
LPC – Linear Predictive Coding;
LFBE – Log Mel-filterbank Energy;
LSTM – Long Short-Term Memory;
MFCC – Mel Frequency Cepstral Coefficients;
NPU – Neural Processing Unit;
OS – Operating System;
PLP – Perceptual Linear Prediction;
RAM – Random-access Memory;
RTF – Real Time Factor;
ROC – Receiver Operating Characteristic;
ReLU – Rectified Linear Unit;
SGD – Stochastic Gradient Descent;
SVM – Support Vector Machine;
TDNN – Time-Delay Neural Network;
TNR – True Negative Rate;
TPR – True Positive Rate;
UML – Unified Modeling Language.

Contents

INTRODUCTION 1
Problem Formulation 1
Relevance of the Thesis 1
The Object of the Research 2
The Aim of the Thesis 2
The Tasks of the Thesis 2
Research Methodology 3
Scientific Novelty of the Thesis 3
Practical Value of the Research Findings 4
The Defended Statements 4
Approval of the Research Findings 5
Structure of the Dissertation 6

1. LITERATURE REVIEW ON VOICE ACTIVATION SYSTEMS 7
1.1. Brief History of Voice Activation System Research 7
1.2. Structure of a Voice Activation System 9
1.2.1. Feature Representation 11
1.2.2. Acoustic Model 15
1.2.3. Acoustic Units 20
1.2.4. Decoding 20
1.2.5. Quality Assessment 23

1.3. Other Approaches 29
1.4. Training a Keyword Spotter in a Low-Resource Dataset Setup 31
1.5. Conclusions of Chapter 1 and Formulation of the Thesis Tasks 34

2. IMPROVING THE QUALITY OF VOICE ACTIVATION BY MODIFYING PARTS OF A SYSTEM, OTHER THAN A NEURAL NETWORK 37
2.1. Investigation of Acoustic Features for a Voice Activation Problem . . . 38
2.1.1. Simplification Trend and Hyperparameters in Acoustic Feature Pipelines 38
2.1.2. Feature Extraction Pipelines Description 39
2.1.3. Comparing Feature Extraction Pipelines and Hyperparameter Investigation 44
2.1.4. Quantitative Results of Acoustic Feature Pipeline Comparison and Hyperparameter Investigation 46
2.1.5. Discussion of Experimental Results for the Acoustic Feature Pipeline Comparison 50
2.2. Investigation of Acoustic Units for a Voice Activation Problem 51
2.2.1. Motivation of the Keyword Decomposition into Acoustic Units 51
2.2.2. Widely Used Acoustic Units 52
2.2.3. Uniform and Adaptive Targets 53
2.2.4. Fast Computation of an “Adaptive” Split 54
2.2.5. Empirical Evaluation of the Acoustic Unit Choice and Hyperparameters 58
2.2.6. Results and Discussion of the Empirical Evaluation of the Acoustic Unit Choice 59
2.3. Detecting Keyword Repeats in Voice Activation Systems 61
2.3.1. Improving the False Reject Rate by Detecting Keyword Repeats 61
2.3.2. Proposed Modification for the Keyword Repeats Detection 63
2.3.3. Proposed Method for the Keyword Repeats Detection 67
2.3.4. Empirical Results of the Proposed Method 69
2.3.5. Discussion of Experimental Results of the Proposed Method . . 73
2.4. Conclusions of Chapter 2 73

3. TRAINING A VOICE-ACTIVATION SYSTEM IN A LOW-RESOURCE SETUP 75
3.1. Collecting Lithuanian Speech Commands Dataset 76
3.2. Empirical Evaluation of the Methods for Training a Keyword Spotter in a Low-Resource Setup 78
3.2.1. Datasets 78
3.2.2. Neural Network Architectures 79
3.2.3. Neural Network Training 81

3.2.4. Baseline Metrics for all Datasets 84
3.2.5. Unsupervised Pre-training for a Low-Resource Setup 84
3.2.6. Fine-Tuning the Pre-Trained Model 87
3.2.7. Exemplar-Like Pre-Training 88
3.2.8. Self-Training 91
3.2.9. Joint Training on Two Datasets of Different Languages 91
3.3. Discussion of Experimental Results of the Investigated Methods 93
3.4. Conclusions of Chapter 3 100

GENERAL CONCLUSIONS 103

REFERENCES 105

LIST OF SCIENTIFIC PUBLICATIONS BY THE AUTHOR ON THE TOPIC OF THE DISSERTATION 121

SUMMARY IN LITHUANIAN 123

ANNEXES 139
Annex A. The Result Tables of Deep Learning Experiments 139
Annex B. Research Information Sheet For Participants 141
Annex C. Informed Consent Form 143

Introduction

Problem Formulation

The algorithm for finding a pre-defined keyword (or keyphrase) in the audio stream is difficult to formulate. Consequently, machine learning and especially deep learning methods are widely used for solving this task. These methods allow inferring the decision rules for classification from the training data.

One of the drawbacks of such an approach is the need for relatively large datasets. For example, Chen et al. (2014a) used hundreds of thousands of samples per keyword and Jose et al. (2020) used millions. While there are several datasets for English (most notably, the Google Speech Commands dataset (Warden, 2018)), there are almost no datasets for low-resource languages, which makes it difficult to check whether state-of-the-art methods generalize well for such languages or low-resource datasets in general.

The improvement of the neural network part of the voice activation system has received much attention in the current research. However, other parts of the system are also important for its resulting quality.

Relevance of the Thesis

Researchers studying the keyword activation problem mostly use complicated machine learning techniques. This often requires large datasets, which are difficult to

collect. Consequently, many researchers test their findings on the well-resourced and well-known English datasets, such as the Google Speech Commands dataset (Warden, 2018). This undoubtedly allows the development of modern voice activation systems; however, such a situation steers researchers to investigate only one side of the problem.

The current research is mainly focuses on improving the neural network part of the system because it often gives the biggest quality boost when the dataset is large. Secondly, most hypotheses are not tested for low-resource datasets.

Investigating methods that give the best results for low-resource setups allows building a higher quality voice activation system for languages like Lithuanian.

The Object of the Research

The object of the present study is the application of deep learning methods for voice activation systems, especially by modifying parts of systems other than a neural network and building such systems with low-resource datasets.

The Aim of the Thesis

The present study aims to improve the quality of voice activation systems by applying deep learning methods, focusing on modifying parts of systems, other than a neural network and on low-resource datasets.

The Tasks of the Thesis

To achieve the thesis aim, the following tasks must be solved:

1. To perform a scientific literature review on the current state of voice activation systems.
2. To evaluate the possibility of using an audio spectrogram feature extraction pipeline due to the simplification trend in acoustic feature pipelines for voice activation systems and investigate hyperparameters of such pipelines.
3. To propose and evaluate new acoustic units that require less a priori knowledge of the target language of a voice activation system and compare them to existing choices.
4. To propose and evaluate a modification of decoding module for phoneme-based voice activation systems for improving the detection of keyword repeats by 10%.

5. To collect a Lithuanian voice command dataset for the evaluation and further research on voice activation systems for setup with less than 20 training examples for each keyword.
6. To propose and evaluate deep learning methods for improving detection accuracy by 7% for a Lithuanian voice activation system in a setup with less than 20 training examples per keyword.

Research Methodology

The quantitative methodology has been used for this thesis. To get a better insight into the possibilities for improvement of the voice activation systems, a literature review of current methods has been conducted. Later, several experiments were executed to collect a novel dataset for building voice activation system in a a low-resource setup and to determine best-performing machine learning and deep learning methods. The computer with Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz processor, 8 GB RAM and Ubuntu 18.04 operation system, and Audacity[1] software was used for dataset collection and some of the experiments. Another computer with Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz, 32 GB RAM, Tesla M40 24GB GPU card, Ubuntu 18.08 operating system and libraries of Keras, Tensorflow, PyTorch, and Kaldi was used for experiments. The gathered data were compared with metrics typical for voice activation systems: accuracy, precision, recall, false accept rate and false reject rate.

Scientific Novelty of the Thesis

The scientific novelty of this study is specified as follows:

- The investigation of acoustic feature pipeline hyperparameters showed that the choice of frame length value of 40 ms and 55 ms for an MFCC acoustic feature pipeline allows a 0.95% ± 0.77% (95% confidence interval) improvement of detection quality compared to the default values of 20 ms and 30 ms;
- The detection accuracy is 11.78% ± 7.77% (95% confidence interval) higher when all the occurrences of the acoustic unit present in a training set are used as targets, compared to the option of using only the occurrences of the acoustic unit inside the keyword;
- The method of detecting 13.32% ± 1.76% (95% confidence interval) of

[1] Audacity® software is © 1999-2020 Audacity Team. The name Audacity® is a registered trademark of Dominic Mazzoni.

previously undetected repeats of the activation while not introducing new false alarms by modifying the existing decoding module of a voice activation system was proposed;

- A novel dataset for the Lithuanian was created to be used for building and evaluating the quality of voice activation systems in a low-resource setup. It includes recordings of 20 keywords, each uttered by 28 people on a mobile phone, and 292 non-speech segments;
- Methods for building a voice activation system with low-resource datasets were proposed and investigated. The best proposed methods showed a relative improvement of 10.17% ± 2.11% (95% confidence interval) for feedforward and residual neural network architectures compared to a standard baseline.

Practical Value of the Research Findings

The achieved results have significance both from the theoretical and practical viewpoints.

The novel dataset for the Lithuanian language will allow researchers to continue experiments for building high-quality voice activation systems for low-resource languages, or at least validate the proposed methods for such cases.

Findings in the modification of parts of voice activation systems other than a neural network show better ways to fine-tune voice activation systems, often without tedious re-training of the neural network. This also highlights the importance of deeper research in this area of keyword spotting.

Finally, the proposed methods of building a voice activation system in a low-resource setup allow for achieving a better voice interaction experience for languages with sparse audio resources.

The Defended Statements

The following statements, based on the results of the present thesis, may serve as the official hypotheses to be defended:

1. The choice of frame length values of 40 ms and 55 ms for in Mel frequency cepstral coefficients feature extraction pipeline of a voice activation system allows to get 0.95% ± 0.77% (95% confidence interval) improvement in detection accuracy for the Google Speech Commands dataset (Warden, 2018) and convolutional neural networks compared to the default values of 20 ms and 30 ms.

2. The detection accuracy is 11.78% ± 7.77% (95% confidence interval) higher when all the occurrences of the acoustic unit present in the training set are used as targets, compared to the option of using only the occurrences of the acoustic unit inside the keyword.
3. The proposed modification of the decoding module in the phoneme-based activation system allows detecting 13.32% ± 1.76% (95% confidence interval) of previously undetected repeats of the activation word without introducing new false alarms. By allowing 1% of new false alarms, 79.9% ± 1.5% (95% confidence interval) of previously undetected repeats can be found. The proposed method requires no re-training of the neural part of the system and introduces less than 4% of additional processing time of the same audio duration in the worst case.
4. The proposed unsupervised pre-training of the audio feature extraction pipeline allows the improvement the accuracy of the voice activation system by 8.04% ± 4.73% (95% confidence interval) when the number of samples per keyword is seven or less and by 24.68% ± 6.24% (95% confidence interval) if the number of samples per keyword is five or less for English, Lithuanian and Russian datasets.
5. The proposed method of joint training on Lithuanian and English datasets shows a relative improvement of 10.17% ± 2.11% (95% confidence interval) for feedforward and residual neural network architectures for the novel Lithuanian dataset and the English Google Speech Commands dataset (Warden, 2018).

Approval of the Research Findings

The results of the dissertation were published in six scientific publications. Three of them were published in refereed journals of *Clarivate Analytics* (also referred to as *Thomson Reuters*) *Web of Science* databases. Two were published in conference proceedings and one in the non-refereed journal. The author also gave four presentations at international and domestic scientific conferences:

- The 11th International Workshop on Data Analysis Methods for Software Systems (DAMSS 2019), 2019, Druskininkai, Lithuania;
- 2020 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream), 2020, Vilnius, Lithuania;
- VI International scientific conference “High Technologies. Business. Society,” 2021, Borovets, Bulgaria;

- 2021 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream), 2021, Vilnius, Lithuania.

Structure of the Dissertation

The scientific work consists of an introduction of the dissertation, three main chapters, general conclusions, references, a list of the author's publications and annexes.

The total scope of the dissertation is 121 pages without annexes.

There are 21 equations, 32 pictures, and 51 tables in the text. 164 references were used in the dissertation text.

1

Literature Review on Voice Activation Systems

This chapter reviews methods used for designing a voice activation system, describes its purposes and important characteristics. The structure of a typical voice activation system is proposed. The main approaches for each module of such a system are described: the choices for acoustic feature extraction pipelines, different types of acoustic models, and decoding methods. Also, the outputs related to building a keyword spotter in a low-resource dataset setup are presented. This chapter concludes with a summary of the literature review findings and the clarification of the main objective and tasks of the present investigation.

The proposed review and the structure of a typical voice activation system were published in an international journal (Kolesau and Šešok, 2020c).

1.1. Brief History of Voice Activation System Research

The voice activation task has been attracting researchers and the industry for decades. Since formulating an algorithm for determining whether a code phrase has been uttered in an audio stream is difficult, 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. The following important moments should be highlighted: the beginning of the use

of the hidden Markov model (HMM) back in 1989 (Rohlicek et al., 1989), the use of neural networks since 1990 (Morgan et al., 1991, 1990; Naylor et al., 1992), the use of pattern matching approaches, in particular dynamic time warping or DTW (Zeppenfeld and Waibel, 1992), optimization of a loss functions specific to a voice activation (as opposed to the common metrics such as accuracy and similar; this enables to get a system more attractive from the point of view of user experience) (Chang and Lippmann, 1994; Szöke et al., 2010), attempts to rid of a garbage model (Junkawitsch et al., 1997), building systems of voice activation for non-English languages such as Chinese (Hao and Li, 2002; Zheng et al., 1999), Japanese (Ida and Yamasaki, 1998), Iranian (Shokri et al., 2011), construction of discriminative systems (Keshet et al., 2009; Tabibian et al., 2011, 2013), publications describing voice activation systems in mass products (Chen et al., 2014a; Gruenstein et al., 2017; Guo et al., 2018; Wu et al., 2018), as well as publishing open datasets to compare different approaches (Warden, 2018).

Voice activation systems are used in various areas: telephony (Shokri et al., 2013; Szöke et al., 2010), crime analysis (Kavya and Karjigi, 2014), the assistance systems in emergency situations (Zhu et al., 2013), automated management of airports (Tabibian, 2017), smart homes (Dogru et al., 2017; Edu et al., 2020) and, naturally personal voice assistants, built-in in mobile phones and home devices (Gruenstein et al., 2017). Many researchers are focused on improving the quality of detection using modern deep learning algorithms and big datasets (Giraldo et al., 2021; Huang et al., 2022; López-Espejo et al., 2021; Ng et al., 2022a,b).

The problem of voice activation is closely related to the problems of automatic speech recognition (ASR) and spoken term detection. In ASR, the task is to find the most likely sequence of words spoken on the audio recording, whereas in voice activation, one needs to find only a predetermined set of words or to indicate that such word/words was/were not spoken. Of course, being able to solve the ASR problem can easily solve the problem of voice activation, but at the moment, most of speech recognition systems consume unacceptably many resources for voice activation.

Spoken term detection is a search for a given phrase (and this phrase may vary depending on the request) in a static set of audio data. In voice activation, the phrase is fixed, but the audio data is delivered in real-time. Therefore, you can use offline methods in spoken term detection, such as bidirectional neural networks or audio pre-indexing.

Despite the differences in these problems, approaches and ideas often overlap. For example, audio data representation, decoding methods, or the architecture of acoustic models.

Additional requirements may apply for voice activation systems. For example, responding only to a keyword that was addressed to the system, but not to the same keyword spoken in the conversation (wake-up-word detection) (Këpuska and Klein,

2009; Siegert et al., 2021; Zhang et al., 2016); responding only to a keyword spoken by a registered user (Gruenstein et al., 2017; Kurniawati et al., 2012; Manor and Greenberg, 2017).

This research is focused primarily on voice activation systems that can be used in embedded systems, particularly in mobile phones. Such systems must satisfy the following properties:

- high recall of finding the keyword (to build a voice interface, one needs to be sure that one can start the voice interaction; with a low recall, the user will have to start an interaction in a different way);
- a small number of false positives (since the voice activation system is always on, many false positives is unacceptable: this wastes device resources, distracts the user's attention, and potentially reduces security);
- the ability to work entirely on a limited resource device (first, continuous forwarding of audio data to remote servers is impossible due to prohibitively high requirements for resources and communication coverage, and second, it is undesirable from the user privacy perspective);
- consumption of little resources (due to the previous property, consuming many resources will lead to rapid battery depletion and slow operation of other processes);
- robustness to noise and variability of a speech;
- a small delay between the utterance of the keyword and system activation.

Systems that satisfy these properties are called *small-footprint keyword activation systems* in this research, similar to Chen et al. (2014a). Thus, some papers that suggest the operation of the system in milder conditions (for example, not in real time) were omitted from the study.

Many recent voice activation systems use specialized hardware like Digital Signal Processor (DSP) or Neural Processing Unit (NPU) (Giraldo and Verhelst, 2021; Guarneri et al., 2022; Ulkar and Okman, 2021). Some use even more specific hardware (Kim et al., 2022). This thesis focuses more on the algorithmic and training parts and situations with under-resourced training data.

Some previous reviews on voice activation systems (Bohac, 2012; Morgan and Scofield, 1991; Rohlicek et al., 1993) contain outdated information (due to the rapid development of the area).

1.2. Structure of a Voice Activation System

As described in Section 1.1, voice activation systems have come a long way. One way to study and compare approaches is to provide the model of a system and to

compare the individual components of the model. Most voice activation systems (especially modern) consist of the following parts:

- *feature extraction* from audio data (to represent audio data in the format acceptable to machine learning models and obtain input data that has enough information to solve the problem);
- application of the *acoustic model* (a system that generally computes the probability of acoustic observations, which often comes down to computing $P(u|O)$, where u is an acoustic unit and O are acoustic observations);
- *decoding* – the process of determining the state sequence with reference to acoustic observation and acoustic model to determine whether a keyword has been uttered or not.

For example, Chen et al. (2014a) described voice activation systems that apply acoustic model specified by a deep neural network to extracted Log Mel-filterbank (feature extraction) and decide whether the keyword was uttered by smoothing deep neural network outputs and comparing them with a threshold (decoding). The illustration of the structure of a typical voice activation system is presented in Fig. 1.1.

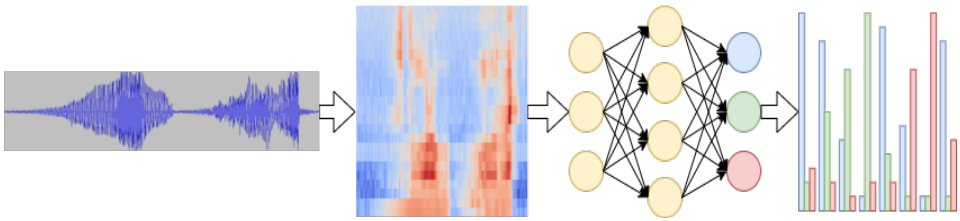


Fig. 1.1. Structure of a typical voice activation system

Of course, not all the possible voice activation systems are well described by the proposed scheme. For instance, in pattern-matching approaches it is hard to separate an acoustic model and feature extraction. Discriminative spotters would be another example. These and other systems are discussed in more detail in Section 1.3. Nevertheless, even in these systems, it is always possible to point out the feature representation of the audio or some kind of an acoustic model.

This literature review aims to summarize the information available in studies about voice activation systems for embedded devices by answering the following research questions:

1. What acoustic features are used?
2. What acoustic model types are used?
3. What acoustic units are used in acoustic modeling?

4. What decoder types are used?
5. What metrics are used to evaluate the systems' quality?

1.2.1. Feature Representation

Sound is a continuous physical phenomenon of mechanical vibrations transmission in the form of an acoustic wave. However, most machine learning models do not accept continuous data as input. Thus, the extraction of features from the audio recording has two main goals:

- representing audio in a way suitable for machine learning methods;
- the preservation of the greatest possible amount of information needed to solve the problem (i.e., finding keywords) and the exclusion of the greatest possible amount of information irrelevant to the task ("noise" such as background sounds or the speech variability).

Most voice activation systems use an approach similar to speech recognition systems (Hinton et al., 2012).

1. The original recording is segmented in possibly overlapping frames.
2. In each frame, a numerical vector that describes the behavior of the sound at this time interval is computed. Usually, this vector is computed using the discrete Fourier transform (DFT). Let's say that this vector has dimension n_{f}.
3. The resulting numerical matrix of the size $T \times n_{\mathrm{f}}$ is used as the result of feature extraction (where T is the number of frames).

Thus the audio data can be viewed as a 2D image or a time series. The specially selected transformation used in the second step is responsible for extracting the most discriminative features for the voice activation task.

Of course, not all systems function in this way. For example, Kumatani et al. (2017) used a raw waveform (without any selected transformations), and Lehtonen (2005) developed a specific digital signal processing pipeline.

Sometimes, feature quantization is used to increase the speed of operation, reduce consumption or for specific algorithms (Feng and Mazor, 1992).

Mel frequency cepstral coefficients (MFCC) are the most frequently used feature type in the studied sources. It is calculated in the following way:

1. The audio is segmented into short frames (the popular choice is to have 25 ms segments with an overlap of 10 ms).
2. For each frame the periodogram estimate of the power spectrum is computed. This is similar to the way human cochlea processes the information (different nerves fire signals depending on the frequency of the audio). To get the

estimate, first, the discrete Fourier transform of each frame is computed via:

$$S_j(k) = \sum_{n=1}^{N} s_j(n)h(n)\exp\left(\frac{-2i\pi}{N}kn\right), \tag{1.1}$$

where j is the frame number, $1 \leqslant k \leqslant K$, K is the DFT length, $h(n)$ is an N sample long analysis window (e.g. Hamming window), $s_j(n)$ is the n-th sample of the j-th frame. Then, the periodogram estimate is computed by:

$$P_j(k) = \frac{1}{N}|S_j(k)|^2. \tag{1.2}$$

3. Apply the Mel-filterbank to the power spectra summing the energy in each filter. The Mel scale relates the perceived frequency. A human ear is much more sensitive to small changes in low frequencies than in higher spectra. To convert frequency f to the Mel scale, the following formula is used:

$$M(f) = 1125\ln(1 + f/700). \tag{1.3}$$

4. The logarithm of filterbank energies is taken. This also relates to human perception: the loudness does not change linearly with the energy. A logarithm is a good approximation, and also, it allows to perform the channel normalization with a simple subtraction (e.g., cepstral mean normalization).
5. A discrete cosine transform is applied. This is done to decorrelate the filterbank energies, which were computed with overlapping filters.

Although most articles use Log Mel-filterbank (sometimes abbreviated as LFBE or log Mel-filterbank energy) or Mel frequency cepstral coefficients or their derivatives, the question arises whether this approach is universal, i.e., suitable for all situations. It turns out that this is not the case. For example, during the development of voice activation systems for the Japanese (Ida and Yamasaki, 1998) prosodic information had to be used to achieve acceptable quality, as MFCC did not render sufficient results. This situation occurs in some other languages (Zheng et al., 1999).

Among the common techniques are stacking (concatenation of acoustic feature vectors from the current and neighboring frames) and the calculation of delta or derivatives (i.e., the calculation of a discrete time derivative using acoustic features from neighboring frames). Also, a mean normalization or a variance normalization are often used. In the cepstral range, this transformation is usually abbreviated as cepstral mean and variance normalization.

Another (slightly less popular) way to extract audio features is perceptual linear prediction (PLP), introduced by Hermansky et al. (1985). This technique is based on three concepts from the psychophysics of hearing:

- the critical-band spectral resolution;
- the equal-loudness curve;
- the intensity-loudness power law.

The spectrum is approximated by a high-order (5-order models are used almost exclusively) autoregressive model. This computationally efficient way to extract physically justifiable audio features was widely used in speech recognition, but is not that popular in modern voice activation systems.

Feature space maximum likelihood linear regression (f-MLLR) is a way of adapting source acoustic features (Gales, 1998). Usually, it is used for a speaker adaptation. When the speaker is not known in advance, the f-MLLR is not very useful, so it is usually not applied in general-use voice activation systems.

Pitch or fundamental frequency is defined as the lowest frequency of a periodic waveform. There is no algorithm for pitch detection that works in all domains. The most popular algorithms are discussed by Gerhard (2003).

As mentioned before, one of the goals of audio feature extractions is to omit as much information as possible about noise and to preserve as much information as possible about for the downstream task or keyword spotting in the case of the current research. One way to achieve this is to try to represent an audio signal with a small number of parameters. If such representation is accurate enough (e.g., the input signal can be restored somewhat accurately from this low-dimension representation), then it might mean that the essential information is preserved in a small number of dimensions. Several methods are based on this logic, namely, linear predictive coding (LPC) (Gish et al., 1990) and linear discriminant analysis (LDA) (Junkawitsch et al., 1997) applied to upstream audio features (e.g., MFCC). To remove the noise component, smoothing transformations can be applied, such as autoregressive moving average (AMA) (Shokri et al., 2011).

The popularity of some acoustic features in studies is presented in Fig. 1.2. The comparison between the popularity in all reviewed studies and the studies presented at the Interspeech 2020 conference shows that Mel frequency cepstral coefficients are still widely used. But at the same time, even simpler acoustic features, such as log Mel-filterbank energy, are gaining popularity, while the need for additional transformations (and prior knowledge of voice generation) is decreasing. It can be hypothesized that it is because modern neural network architectures and the spread of large volume datasets make learning patterns from data more profitable than introducing more information via careful feature engineering. It will be discussed in more detail in Section 2.1, where the experiments of simplifying the acoustic feature pipelines will be performed.

Another way to get acoustic features is to extract them in an unsupervised manner. This approach will be discussed in Chapter 3.

The summary information of used acoustic features and their transformations in reviewed studies is presented in Table 1.1.

Table 1.1. Acoustic features used in the studied sources

Acoustic features and transformations	Sources
MFCC	Bahi and Benati (2009); Baljekar et al. (2014); Bluche and Gisselbrecht (2020); Fernández-Marqués et al. (2018); Heracleous and Shimizu (2003); Jansen and Niyogi (2009c); Junkawitsch et al. (1997); Keshet et al. (2009); Khne et al. (2004); Laszko (2016); Leow et al. (2012); Li et al. (2020b); Liu et al. (2000); Manor and Greenberg (2017); Mo et al. (2020); Rose and Paul (1990); Rybakov et al. (2020); Sangeetha and Jothilakshmi (2014); Shokri et al. (2014, 2013); Szöke et al. (2005); Tabibian et al. (2011); Vasilache and Vasilache (2009); Vroomen and Normandin (1992); Wöllmer et al. (2009a,b, 2013); Xu and Zhang (2020); Yilmaz et al. (2020); Zehetner et al. (2014); Zhu et al. (2013)
LFBE	Chen et al. (2014a); Gruenstein et al. (2017); Hou et al. (2016); Hwang et al. (2015); Liu et al. (2020); Lopatka and Bocklet (2020); Morgan et al. (1991, 1990); Myer and Tomar (2018); Sun et al. (2017); Wu et al. (2020); Zeppenfeld and Waibel (1992); Zhang et al. (2020); Zhang and Zhang (2020)
Energy or log-energy	Heracleous and Shimizu (2003); Hwang et al. (2015); Khne et al. (2004); Vroomen and Normandin (1992); Wöllmer et al. (2013)
Mean/variance normalization	Gish and Ng (1993); Gish et al. (1992); Jansen and Niyogi (2009c); Myer and Tomar (2018); Rohlicek et al. (1993); Sangeetha and Jothilakshmi (2014); Shokri et al. (2011); Wöllmer et al. (2009a)
Gain normalization	Shokri et al. (2011)
Stacking	Chen et al. (2014a); Fernández-Marqués et al. (2018); Junkawitsch et al. (1997)
LDA over MFCC	Junkawitsch et al. (1997)
Fourier transform	Guo et al. (2018); Morgan et al. (1991, 1990); Zeppenfeld and Waibel (1992)
LPC	Gish et al. (1990); Gish and Ng (1993); Gish et al. (1992); Rohlicek et al. (1993, 1989); Zheng et al. (1999)

End of Table 1.1

Acoustic features and transformations	Sources
Prosodic information	Ida and Yamasaki (1998); Zhang et al. (2022)
PLP	Chen et al. (2014a); Ge and Yan (2017); Szöke et al. (2005)
AMA	Shokri et al. (2011)
Spectral entropy, spectral flatness, burst degree, bisector frequency	Tabibian et al. (2011)
Formant frequencies	Laszko (2016)
f-MLLR	Sadhu and Ghosh (2017)
Raw waveform	Kumatani et al. (2017)

A detailed description of the mentioned features is given in relevant articles or reviews (Giannakopoulos, 2015). The visualization of some of the features for the phrase "Hello, world!" are shown in Fig. 1.3 and computed using the framework for speech recognition Kaldi (Povey et al., 2011).

1.2.2. Acoustic Model

The task of the acoustic model is to model theacoustic properties of the selected acoustic unit. For example, an acoustic model can provide a probability distribution over the vectors of MFCC features when a certain word is pronounced. Practically, the acoustic model is used to compute $P(S|u)$, where S – sound and u is some acoustic unit.

Often, it is more natural or easier to compute $P(u|S)$, and then get $P(S|u)$ via Bayes theorem 1.4. Especially often, this technique is used in conjunction with a hidden Markov model (HMM):

$$P(S|u) = \frac{P(u|S)P(S)}{P(u)}, \tag{1.4}$$

where u is an acoustic unit (in practice, one can think of HMM state); S is a sound (which can be represented, for example, like an MFCC feature vector); $P(S|u)$ is a posterior probability of the generation of the given sound (e.g., MFCC feature vector) S upon the speaker pronouncing the acoustic unit (e.g., a phoneme) u at the given moment (alternatively, the HMM is in the state u); $P(u|S)$ is a likelihood of

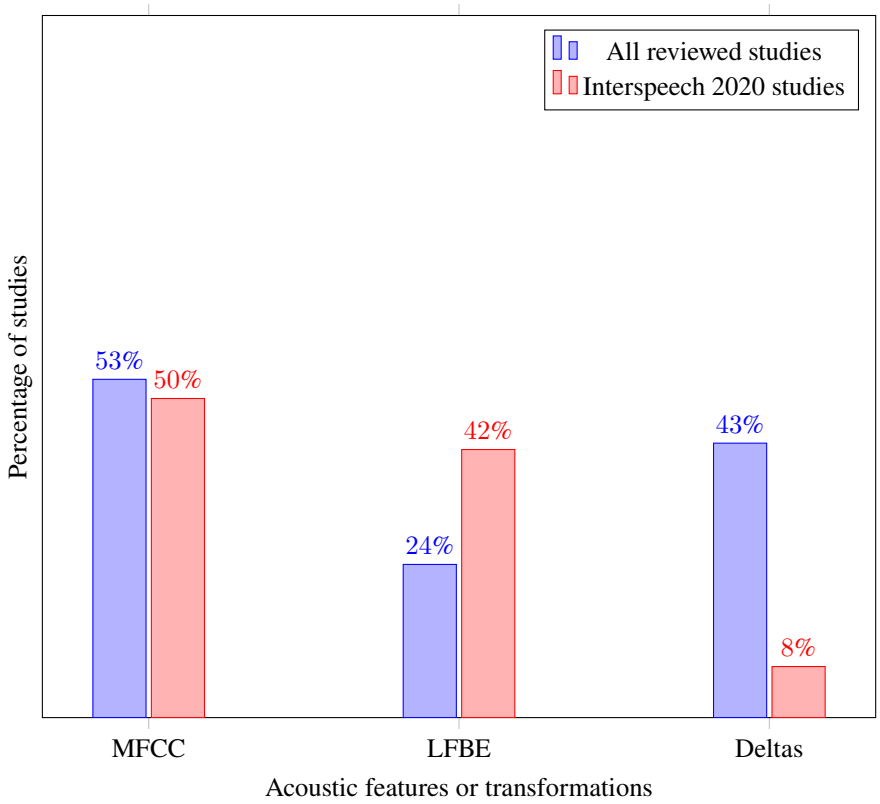


Fig. 1.2. Percentages of studies that use some popular acoustic features or transformations

the event of pronouncing the acoustic unit u (or being in the HMM state u) given the sound (e.g., MFCC feature vector) S, which is usually computed by some acoustic model; $P(u)$ is a prior probability of pronouncing the acoustic unit u (which can be estimated by a separate language model, like in ASR (Hinton et al., 2012)) and $P(S)$ is a prior probability of sound S. Note that $P(S)$ might be difficult to compute, but it is not necessary, at least in an HMM problem statement: the S is fixed for a given audio stream, so every choice of u has a constant $P(S)$ coefficient, which does not influence the maximal likelihood choice.

The most common acoustic model for voice activation is built as follows. The set of HMM states is logically divided into two parts: a part that represents an audio event of keyword pronunciation and a garbage model (a model of the rest of the sound: noise, background speech, and actual voice request). Fig. 1.4 shows a typical HMM used in Amazon's spotter for the keyword "Alexa" used in work by Guo et al. (2018).

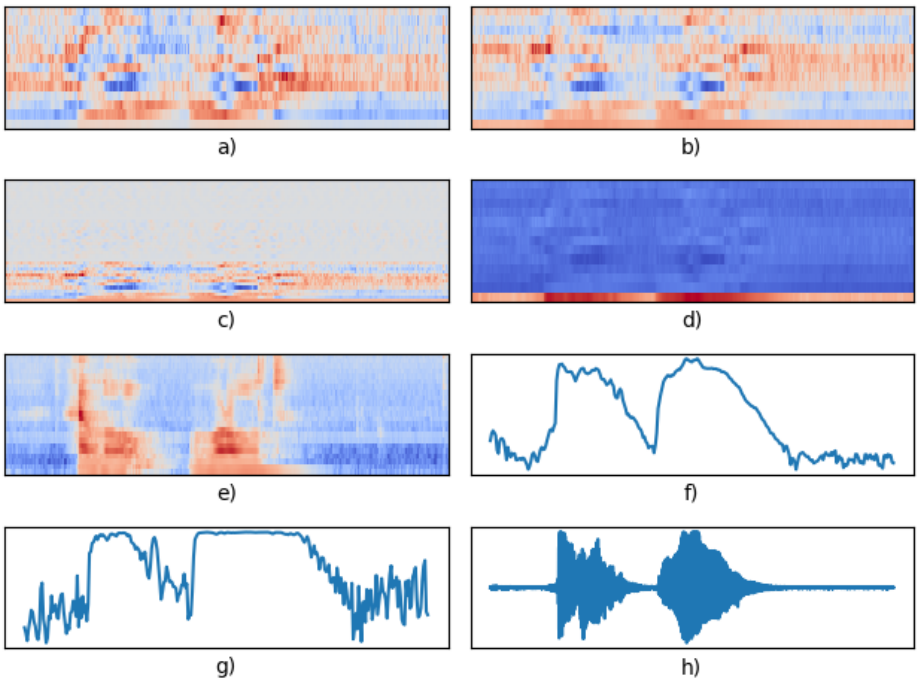


Fig. 1.3. Feature visualization for the audio file with "Hello, world!" pronunciation: a) MFCC with CMVN; b) MFCC; c) MFCC with delta; d) PLP; e) LFBE; f) energy; g) pitch; h) raw waveform

Each state of the model represents an acoustic unit (see Section 1.2.3 for details), for example, a phoneme. The model "says" that at each frame (Section 1.2.1), the acoustic environment is in one of the states of the HMM, and it generates visible variables, for example the vector of the MFCC features (or, more generally, sound). Each state has a distribution of probabilities over the sound. Thus, when an audio file is received, the sound and the probability distributions are known, but the particular states of the model on each of the frames are not known. However, for each possible sequence of states, the probability of this sequence can be calculated. By decoding (Section 1.2.4), the most probable sequence can be found. If this sequence generated a keyword, then it can be said that the activation occurred (there are other options for decoding and determining the activation).

It is necessary to be able to calculate $P(S|s)$ (s is an HMM state) to find the keyword. Such calculation is called acoustic modeling. Gaussian mixture model (GMM) or neural networks are the most frequent choices for an acoustic model. Note that these choices coincide with the choices for acoustic models in automatic

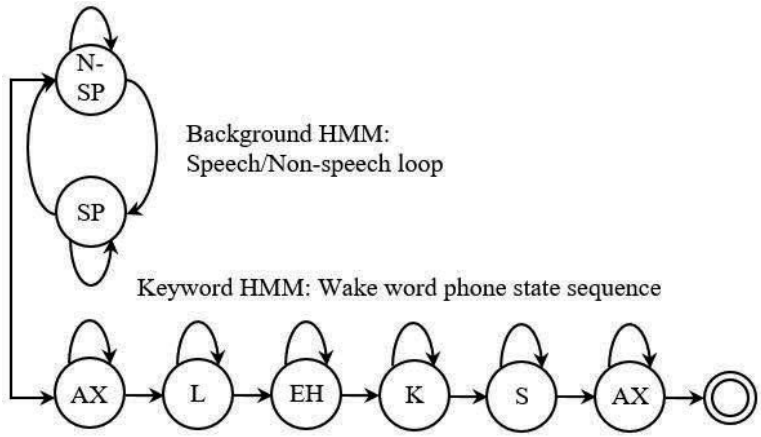


Fig. 1.4. HMM example for the Amazon's keyword spotter (Guo et al., 2018)

speech recognition systems. Before (Hinton et al., 2012) GMM acoustic models were considered state-of-the-art, and after the publication they were almost completely replaced by neural networks.

Note that it is the question of definitions of what to consider an acoustic model in the HMM-GMM setup. You can either consider GMM (the part which actually computes $P(S|s)$, recall that HMM state often represents some acoustic unit) or the whole HMM because it expresses $P(S|w)$ as explained by Zheng et al. (1999) (w is a keyword).

A good acoustic model is key for a high-quality voice activation system. Therefore, it is not surprising that the calculations associated with the acoustic model usually take the biggest part of the voice activation system runtime. Consequently, many effors are trying to speed up this part. For example, Fernández-Marqués et al. (2018) applied the binary arithmetic (instead of the floating arithmetic) in the model, Sun et al. (2017) represented the architecture of a neural network where each layer of the matrix multiplication of $N \times M$ was replaced by the product of two matrices with sizes $N \times K$ and $K \times M$, where K was much smaller than N and M. Thus, a big number of operations was saved, and not a lot of expressive power of the model was lost (with the appropriate method of training).

Another way to build a speech recognition system is not to use HMM, but to calculate some (heuristically selected) value based on the outputs of the acoustic model (Chen et al., 2014a).

The acoustic models used in the studied sources are presented in Table 1.2. The number of times these models were used in the sources is presented in Table 1.3.

Table 1.2. Acoustic models used in the studied resources

Acoustic model	Sources
NN	Chen et al. (2014b); Ge and Yan (2017); Gruenstein et al. (2017); Hou et al. (2016); Lehtonen (2005); Morgan et al. (1990); Myer and Tomar (2018); Szöke et al. (2010, 2005); Wu et al. (2018)
GMM	Baljekar et al. (2014); Benisty et al. (2018); Chen et al. (2014a); Heracleous and Shimizu (2003); Jansen and Niyogi (2009a,c); Junkawitsch et al. (1997); Khne et al. (2004); Leow et al. (2012); Li and Wang (2014); Liu et al. (2000); Rohlicek et al. (1989); Rose and Paul (1990); Shokri et al. (2011); Szöke et al. (2005); Vasilache and Vasilache (2009); Vroomen and Normandin (1992); Zhu et al. (2013)
RNN	Baljekar et al. (2014); Naylor et al. (1992)
Gated RNN	Baljekar et al. (2014); Hou et al. (2016)
TDNN	Kumatani et al. (2017); Myer and Tomar (2018); Sun et al. (2017); Zeppenfeld and Waibel (1992)
Polynomial model	Gish and Ng (1993)
Continuous density neural tree	Kosonocky and Mammone (1995)
Mixture of central distance normal distributions	Zheng et al. (1999)
LSTM	Hou et al. (2016); Hwang et al. (2015)
Bi-LSTM	Wöllmer et al. (2009a); Zhang et al. (2016)
SVM	Tabibian et al. (2011)
DNN with highway blocks	Guo et al. (2018)
Binary DNN	Fernández-Marqués et al. (2018)
CNN	Myer and Tomar (2018)

Table 1.3. Number of times specific acoustic model was used in the studied sources

Acoustic model	Number of sources
GMM	18
Neural network	10
Time-delayed neural network	4
RNN, Gated RNN, LSTM, Bidirectional LSTM	2
Polynomial model, continuous density neural tree, a mixture of central distance normal distributions, support vector machine, Deep neural network with highway blocks, binary deep neural network, convolutional neural network	1

1.2.3. Acoustic Units

The choice of an elementary unit for acoustic modeling (acoustic unit) affects the resulting quality. A system developer is faced with the following tradeoff: the larger is a unit (e.g., a word), the more stable it is (meaning that produced acoustic features have less variability), and accordingly, it is easier to find such a pattern in an audio stream. However, such a system is not flexible.

If a smaller unit has been chosen, for example, a phoneme, then the system is faced with a more difficult task of finding a pattern, but on the other hand, such a system can find an arbitrary word built from the same phoneme set.

Sometimes, the solution for this tradeoff is to choose syllables or parts of the words if it's difficult to define syllables.

Also, it is not necessary to choose the whole phoneme as a unit, but as a part of the phoneme (for example, the beginning of the phoneme A or the middle of the phoneme B) or context-dependent phoneme (for example, phoneme A, going after phoneme B). A phoneme without a context is often called a monophone, and a context-dependent one is called a biphone (if the dependency is only on one side) or triphone (if the dependency is on the left and right). There is also a possibility to combine these approaches and use a part of the context-dependent phoneme. In this case, a system will probably have impractically many units, so they will often be clustered (by pronunciation) into senones.

It must be noted that the term senone does not have a strict definition. Some authors, like Yu and Deng (2014), define a senone as a tied (clustered) triphone state. Others, like the authors of Janus Toolkit refer to all acoustic units as senones (Rogina and Waibel, 1995).

The solution to this tradeoff depends on the size of the training data (at a small size, it is much more difficult to build a whole word model than a phoneme model), the choice of the acoustic model, the key phrase, and the language. To the author's best knowledge, now, there is no algorithm or rules under what conditions which acoustic unit to choose.

The acoustic units used in the studied sources are presented in Table 1.4. The number of times these units were used in the sources is presented in Table 1.5.

1.2.4. Decoding

After applying an acoustic model to an audio stream, the values are received, characterizing the probability that at a certain moment, one or another acoustic unit was pronounced. The voice activation system needs to decide whether the keyword was uttered in an audio stream or not according to the obtained one or more numeric series. To do this, different approaches to decoding are used.

Table 1.4. Acoustic units used in the studied sources

Acoustic unit	Sources
Whole word	Baljekar et al. (2014); Chen et al. (2014a); Cuayáhuitl and Serridge (2002); Fernández-Marqués et al. (2018); Hou et al. (2016); Manor and Greenberg (2017); Morgan et al. (1991, 1990); Myer and Tomar (2018); Naylor et al. (1992); Rohlicek et al. (1993); Rose and Paul (1990); Zehetner et al. (2014)
Monophone	Cuayáhuitl and Serridge (2002); Gruenstein et al. (2017); Heracleous and Shimizu (2003); Hou et al. (2016); Jansen and Niyogi (2009a,c); Kumatani et al. (2017); Lehtonen (2005); Myer and Tomar (2018); Rohlicek et al. (1993); Rose and Paul (1990); Shokri et al. (2011); Silaghi and Vargiya (2005); Szöke et al. (2010, 2005); Tabibian et al. (2011, 2018); Wöllmer et al. (2009a,b)
Triphone	Rose and Paul (1990); Szöke et al. (2005)
Part of the word	Chen et al. (2014a); Li and Wang (2014); Naylor et al. (1992)
State unit	Zeppenfeld and Waibel (1992)
Part of the phoneme	Kosonocky and Mammone (1995); Leow et al. (2012); Rohlicek et al. (1989)
Syllable	Cuayáhuitl and Serridge (2002); Hou et al. (2016); Klemm et al. (1995); Liu et al. (2000); Zheng et al. (1999)
Letter	Hou et al. (2016); Hwang et al. (2015); Lengerich and Hannun (2016)
Senone	Ge and Yan (2017)

Table 1.5. Number of studied sources that used acoustic unit

Acoustic unit	Number of sources
Monophone	19
Whole word	13
Syllable	5
Letter, part of the word, part of the phoneme	3
Triphone	2
State unit (learnt “phoneme”), senone	1

In the simplest case, it is only necessary to compare the obtained number with the threshold value to decide. E.g., when the acoustic unit is the whole keyword, the decision is made by comparing the computed probability with 0.5. Consider the case of two outcomes: the “keyword was uttered,” and “keyword was not uttered”. The stochastic methods compute the probability of different outcomes, and the decision is made by choosing the outcome with the maximum probability. But when there are only two outcomes, this is equivalent to comparing the probability to 0.5.

Smoothing is usually used to improve the recognition quality in the case of comparison with the threshold (Chen et al., 2014a; Lehtonen, 2005). The motivation for this technique is that a keyword is an acoustic event that has a certain duration in the time dimension. Thus, the actual keyword utterance should generate a high probability of multiple counts in a row. Thus, when applying the smoothing function to the time series, false positives caused by fluctuations of the acoustic model are avoided. Silaghi and Vargiya (2005) suggested an interesting variant of smoothing. In the case of acoustic units, the probabilities of each phoneme are normalized to the probability of the least probable phoneme.

In systems that use a comparison with a template utterance, dynamic time warping (DTW) is often used. DTW is an analog of the Levenshtein distance for numerical series. The motivation of this method is that the duration of the recorded pattern is likely to differ from the pronunciation in real conditions. Thus, two audio fragments can not be compared directly, namely, one needs "to stretch" or "to squeeze" certain intervals of the template over time. The DTW distance is usually computed with dynamic programming. For a more detailed description and various modifications, please refer to Zehetner et al. (2014).

Decoding becomes more meaningful in the case of HMM-based voice activation systems. Indeed, in this formulation, a typical problem for HMM needs to be solved: find the most probable sequence of hidden states (if this sequence corresponds to a keyphrase, then, in some approaches, it means activation) or find the total probability of passing through some sequences of states.

The Viterbi algorithm uses dynamic programming to find the most likely sequence of hidden states in HMM, given the observations. Naturally, this algorithm is widely used in works on HMM-based voice activation systems. Many authors explore a variety of approaches and heuristics to speed up the algorithm, adapt it to find sequences satisfying some additional properties, etc. For example, Liu et al. (2000) used various techniques of hypotheses pruning and rescoring probabilities using a bi-gram language model. Zhu et al. (2013) considered the possibility of using the Viterbi algorithm on sliding windows of the audio stream. Junkawitsch et al. (1997) considered a modification of the Viterbi algorithm that approximates finding the optimal sequence that has the highest probability normalized by the utterance length. Several additional modifications of the Viterbi algorithm were considered by Wilcox and Bush (1992).

In addition, Wilcox and Bush (1992) discuss how to use the forward-backward algorithm for quick estimation of probabilities needed in decoding.

The standard technique in using HMM-derived probabilities and deriving decoding for comparing to the threshold should also be mentioned. This approach is conventionally called the likelihood ratio. Often, two HMMs are used: the speech model representing all the keyword pronunciations, and the garbage model representing all other audio events (e.g., Fig. 1.4). In such systems, one can find the

probability of passing through the garbage model and the probability of passing through the part with the keyword. Then, the ratio of these two probabilities shows confidence in the presence of a key phrase in the audio stream. This ratio is compared with the threshold in many voice activation systems. It is worth noting that finding the balance of coefficients in such models is a difficult task, which is usually solved by optimizing the parameters on the held-out data set.

Some authors use completely different approaches to decoding. For example, Manor and Greenberg (2017) described an application of fuzzy logic to decoding.

The approaches to decoding used in the studied sources are presented in Table 1.6. The numbers of times the specific approach was used in the studied sources are presented in Table 1.7.

Table 1.6. Approaches to decoding used in the studied sources

Decoding approach	Sources
Comparing to threshold	Benisty et al. (2018); Chen et al. (2014a); Gruenstein et al. (2017); Junkawitsch et al. (1997); Keshet et al. (2009); Li and Wang (2014); Morgan et al. (1990); Myer and Tomar (2018); Naylor et al. (1992); Wöllmer et al. (2009a,b)
Viterby	Feng and Mazor (1992); Ge and Yan (2017); Junkawitsch et al. (1997); Knill and Young (1996); Kumatani et al. (2017); Leow et al. (2012); Liu et al. (2000); Rohlicek et al. (1993); Rose and Paul (1990); Sun et al. (2017); Tabibian et al. (2011); Vasilache and Vasilache (2009); Wilcox and Bush (1992); Zheng et al. (1999); Zhu et al. (2013)
Forward-Backward algorithm	Rohlicek et al. (1993); Wilcox and Bush (1992)
DTW	Hou et al. (2016); Kosonocky and Mammone (1995); Kurniawati et al. (2012); Zehetner et al. (2014); Zeppenfeld and Waibel (1992)
Likelihood ratio	Jansen and Niyogi (2009c); Szöke et al. (2010)
Fuzzy logic	Manor and Greenberg (2017)

1.2.5. Quality Assessment

Many metrics can be used to compare different approaches to voice activation systems. These metrics can be grouped by the aspect of the system they measure:

- classification quality;
- operation speed;
- amount of used random-access memory and central processing unit.

Table 1.7. Number of times the specific approach to decoding was used in the studied sources

Decoding approach	Number of sources
Viterbi	15
Comparing to threshold	11
DTW	5
Forward–Backward algorithm, likelihood ratio	2
Fuzzy logic	1

Metrics for speed measurement are standard and non-specific for voice activation systems. The most commonly used is the real time factor (RTF) – the total processing time of the audio stream divided by the length of the stream, latency (average delay of the response signal from pronouncing), and total processing time. The last metric is less indicative than RTF because it depends on the length of the dataset, so total processing times can not be compared for different audio streams. Another reason is that total processing time also accounts for operations that are not done by voice activation systems, for example, the operating system (OS) operations.

For resource usage, it is the most popular to measure the amount of RAM used and the CPU load (as a percentage of the compute core). To improve both parameters, different approaches to quantize the parameters of the acoustic model are often used (Fernández-Marqués et al., 2018). Quantization is the process of mapping input values from a large set to output values in a smaller set, often with a finite number of elements. In deep learning, quantization usually refers to using smaller data types to represent neural network parameters.

But now, there are no standard metrics to measure the quality of classification. Moreover, similar metrics, unfortunately, are called differently in different sources. It would be advantageous to have a standardized set of metrics in that area.

The main problem is that the voice activation system must satisfy two opposite properties to work well: it must be sensitive enough to react to the keyword utterances, and it must be robust enough not to react to sound events similar to but not actual keywords. Any system can be made arbitrarily sensitive, reacting to each event, and arbitrarily robust, not reacting to any events. The challenge is to choose the right balance between these two operating points. Therefore, one must either use at least two metrics (for example, precision and recall) or use one common metric (for example, F_1-score) to measure the quality of classification. In the second case, an unsuccessful choice of metrics can lead to false conclusions since there is no single correct balance between the importance of sensitivity and robustness.

The following metrics are often used to measure classification quality:

- detection rate (precision) is the number of correctly recognized keywords relative to the total number of accepted keywords;

- substitution rate is the number of misrecognized keywords relative to the total number of accepted keywords;
- deletion rate (false reject rate (FRR), opposite to recall, miss rate) is the number of undetected keywords relative to the total number of keywords;
- rejection rate is the number of keywords rejected relative to the total number of keywords (FRR);
- false alarm rate (FAR) or false positive rate (FPR) is the number of false alarms (relative to the number of utterances without keyword; sometimes per keyword or per hour of speech);
- accuracy (recognition rate) is the number of correctly classified utterances relative to the total number of utterances;
- true positive rate (TPR) (same as recall);
- true negative rate (TNR) (opposite to FRR).

Thus, there is no accepted pair of metrics; moreover, often, the same metrics are not called the same in different sources.

The figure of merit (FOM) is one of the most used metrics in voice activation system research. FOM is the average of correct detections per k false positive activations per hour for each natural number k from 1 to 10. This metric was especially often used until 2010. Recently, such rates of false positives per hour are unreasonably high, so FOM does not reflect the relevant modes of operation of the modern voice activation system. Another common metric is an equal error rate (EER) (the smallest value that can take both FAR and FRR at the same time). Note that this metric characterizes both false alarms and false rejects, so the single value can be used to compare two different models. The drawback of this metric is that the operating point of equal false alarm and false reject rate might not be optimal for real use, so the comparison might not be relevant.

Another approach would be to try and compare all operating points of one model with all operating points of another model. To do so, the plot of a receiver operating characteristic (ROC) can be drawn. See the example in Fig. 1.5 where ROC curves are drawn: curves that represent all operating points of the model by a set of points with coordinates (FAR, TPR). In the example, “Model A” is undoubtedly better than all the other models because, for every fixed value of FAR, its value of TPR is higher than the values of “Model B,” “Model C,” or the “random guess model.” At the same time, it can not be unambiguously decided which model is better, “Model B” and “Model C,” because there are areas where the latter works better and vice versa. Though there is no unique decision, ROC curves can help if the voice activation system designer has some performance constraints, e.g., if the system must have a TPR of at least 80%, then “Model B” is better than “Model C.” The natural baseline for a ROC curve is the “random guess model” that gives a positive answer with the

probability p regardless of the audio sample. It can be seen that both FAR and TPR will be equal to p for all values of p.

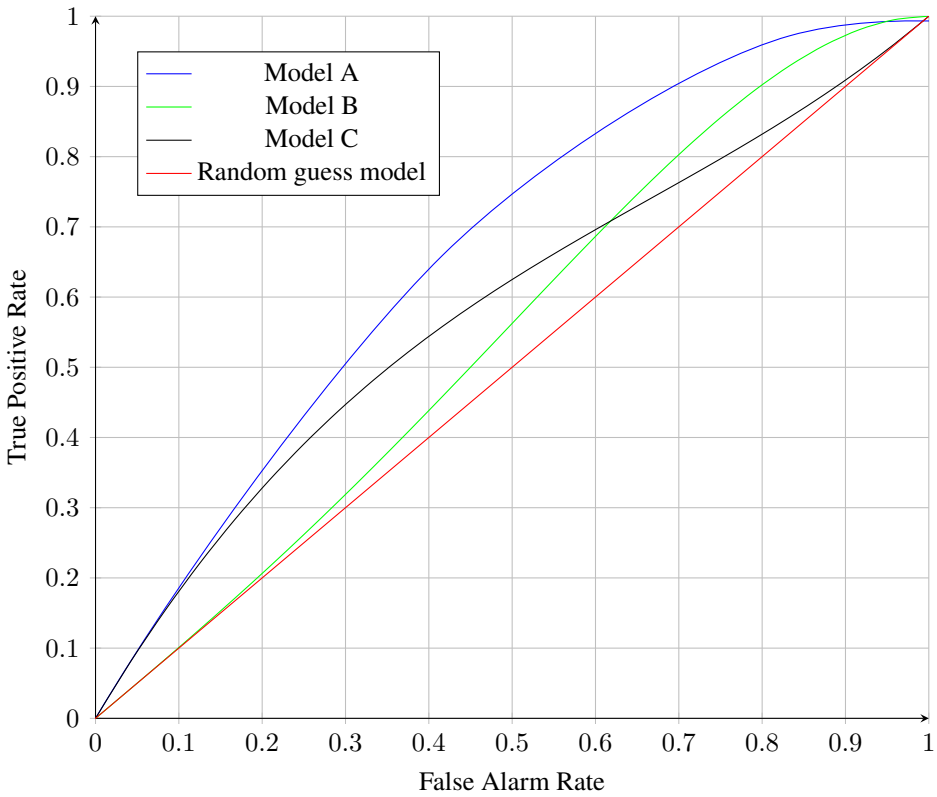


Fig. 1.5. Example of ROC-curves for some hypothetical models and the model which makes a random guess for each audio sample

The natural way to compare two ROC curves is to compute the areas under the curves (AUC). See Fig. 1.6 for example. If one model is better than another for all values of FAR, then the ROC-AUC of that model will also be higher. Note that the opposite is not always true: the higher value of ROC-AUC does not necessarily mean that one model is better than another. For example, “Model B” has a slightly lower value of ROC-AUC than “Model C” but works better for operating points with TPR higher than 80%. Note that ROC-AUC for a random guess model is equal to 0.5 and the best possible value of ROC-AUC is equal to 1.

Some papers suggest more complex ways of measuring classification quality. For example, a discriminative error rate was introduced by Cuayáhuitl and Serridge (2002). In this metric, different errors (when the system asked the user for confirma-

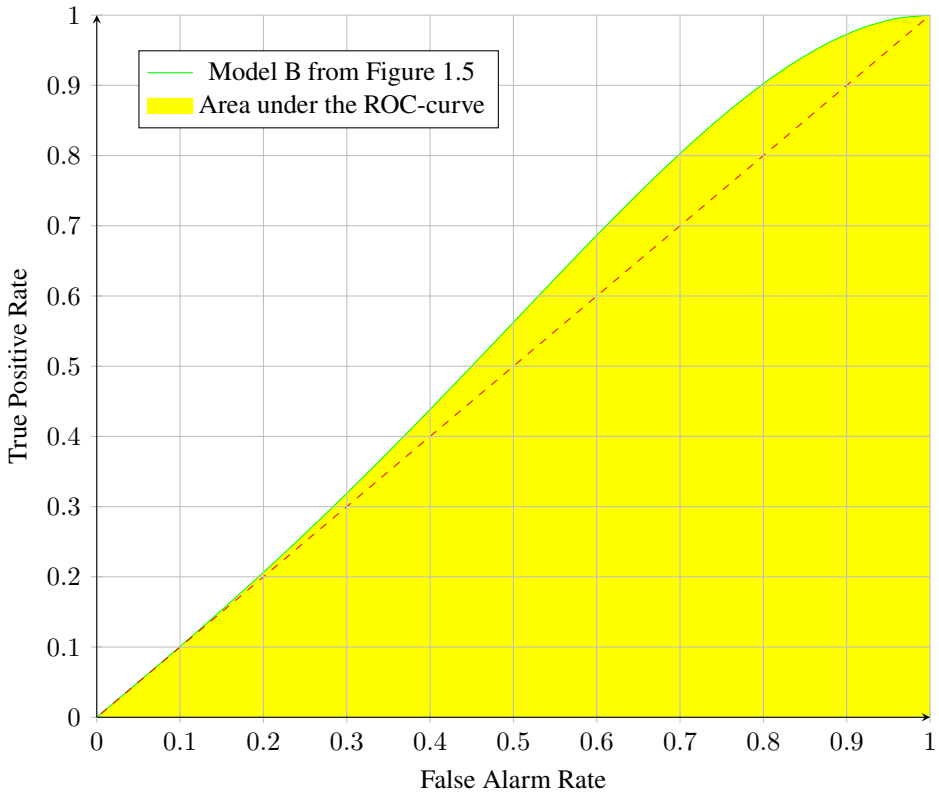


Fig. 1.6. Area under the ROC-curve

tion or rejected the operation without confirmation) have different penalties. This approach is more suitable for quality assessment for a real product use of systems.

It is often quite complicated to compare results from different papers because the choice of the dataset and the keyword deeply affect the results. If two outputs use false alarms per hour metric to describe their system quality, but one uses a dataset of speech recordings, and the other uses a dataset from real user devices (where speech may take 3–6 hours for each 24-hour recording), then these works would have completely different metric values even with the exact same voice activation system.

It is safe to assume that the industry research provides the best or close to the best voice activation systems today because of a large amount of audio data and computation resources. Shan et al. (2018) reported a system with 1.02% FRR with one false alarm per hour. This model has 84 000 parameters. Alvarez and Park (2019) claimed that their “Ok Google” voice activation systems has FRR from

0.87% (clean non-accented utterances) to 8.90% (real user query logs) with 0.1 false alarms per hour with 700 000 parameters. Fernández-Marqués et al. (2018) maintained that it was possible to create a competitive voice activation system using 15.8 kB of memory and performing 2 million operations per inference pass.

The metrics used in the studied sources are presented in Table 1.8. The number of times these metrics were used in the studied sources is presented in Table 1.9.

Table 1.8. Metrics used in the studied sources

Metrics	Sources
FOM	Bohac (2012); Chang and Lippmann (1994); Gish et al. (1990); Gish and Ng (1993); Jansen and Niyogi (2009a,c); Junkawitsch et al. (1997); Knill and Young (1996); Lehtonen (2005); Naylor et al. (1992); Rohlicek et al. (1993); Rose and Paul (1990); Sadhu and Ghosh (2017); Sangeetha and Jothilakshmi (2014); Szöke et al. (2010, 2005); Tabibian et al. (2011, 2018); Zeppenfeld and Waibel (1992); Zheng et al. (1999)
EER	Bohac (2012); Szöke et al. (2010)
Accuracy	Benisty et al. (2018); Fernández-Marqués et al. (2018); Ge and Yan (2017); Ida and Yamasaki (1998); Morgan et al. (1991, 1990)
FA/kW/h	Feng and Mazor (1992); Kavya and Karjigi (2014); Leow et al. (2012); Rohlicek et al. (1989); Vroomen and Normandin (1992)
ROC	Keshet et al. (2009); Kumatani et al. (2017); Marcus (1992); Sadhu and Ghosh (2017); Shokri et al. (2013); Siu et al. (1994); Wöllmer et al. (2009b, 2013)
Detection rate	Feng and Mazor (1992); Khne et al. (2004); Leow et al. (2012); Shokri et al. (2011)
Substitution rate	Feng and Mazor (1992)
Deletion rate	Feng and Mazor (1992); Kavya and Karjigi (2014)
Rejection rate	Feng and Mazor (1992); Heracleous and Shimizu (2003)
Insertion rate	Klemm et al. (1995)
Recognition rate	Heracleous and Shimizu (2003); Liu et al. (2000); Zhu et al. (2013)
Discriminative error rate	Cuayáhuitl and Serridge (2002)
FAR	Benisty et al. (2018); Chen et al. (2014a); Ge and Yan (2017); Gruenstein et al. (2017); Guo et al. (2018); Hou et al. (2016); Khne et al. (2004); Myer and Tomar (2018); Shokri et al. (2011); Sun et al. (2017); Tabibian et al. (2018); Wu et al. (2018)
FRR	Chen et al. (2014a); Gruenstein et al. (2017); Guo et al. (2018); Myer and Tomar (2018); Sun et al. (2017); Wu et al. (2018)
RTF	Bohac (2012); Szöke et al. (2005); Tabibian et al. (2018)

End of Table 1.8

Metrics	Sources
Miss rate	Hou et al. (2016)
Recall	Baljekar et al. (2014); Hwang et al. (2015); Li and Wang (2014); Zehetner et al. (2014)
Precision	Hwang et al. (2015); Zehetner et al. (2014)
F_1, latency	Hwang et al. (2015)
Mean time between false alarms	Baljekar et al. (2014)
Processing time	Li and Wang (2014)
Misses, hits	Li and Wang (2014)
RAM usage, flops, accuracy to size, accuracy to ops	Fernández-Marqués et al. (2018)
Custom	Marcus (1992); Silaghi and Vargiya (2005); Szöke et al. (2010)

Table 1.9. Number of times the specific metric was used in the studied sources

Metrics	Number of sources
FOM	20
False alarm rate	12
ROC	8
FRR, accuracy	6
False alarm per kw per hour	5
Detection rate, recall	4
Custom, recognition rate, RTF	3
EER, deletion rate, rejection rate, precision	2
Insertion rate, discriminative error rate, substitution rate, TPR, FPR, miss rate, F_1, latency, mean time between false alarms, processing time, misses, hits, RAM usage, flops, accuracy to size, accuracy to ops	1

1.3. Other Approaches

Some approaches to the construction of voice activation systems are difficult to describe according to the classification proposed in the Section 1.2.

First, it is worthwhile to mention approaches of comparing with a template, for example, using DTW. In such systems, the user first records one or several keyword pronunciations, and then the necessary sound fragments are compared with the recordings, and the triggering is announced if the selected similarity measure exceeds some prespecified threshold. The advantages of this approach include the simplicity of learning (memorization) and operation. In addition, in this approach, it is natural to use personalization: indeed, one can argue that recorded patterns reflect the specific features of the user pronunciation, which allows to distinguish it from other users if an appropriate similarity metric is used. However, in practice, this approach is not very robust. The quality of its operation depends on how well the similarity measure is chosen and what features are used. The task of eliminating all the noise and dissimilarity in environments by appropriate choice of features and the similarity measure has proven to be difficult. DTW is one way to calculate the measure of similarity of two time series, possibly of different lengths. Systems using such approaches are described by Morgan et al. (1991), Naylor et al. (1992), Zeppenfeld and Waibel (1992), Kosonocky and Mammone (1995), and Kurniawati et al. (2012). Zehetner et al. (2014) discussed different underlying metrics of the similarity to use in the DTW framework. Szöke et al. (2015) discussed the possibility of using a DTW even for the case where a keyword can be subjected to declensions, conjugations, or even word order permutations.

Another interesting approach is to model the appearance (or absence) of keywords with the help of point processes and, in particular, Poisson processes. In such systems, the parameters of two process families are evaluated: for each selected feature for sound with (1) and without a keyword (2). An interesting feature of such systems is the ability to select these parameters during operation, thereby adapting to the channel, user, and usage scenarios. For more information on the proposed, see Jansen and Niyogi (2009b). Sadhu and Ghosh (2017) described how to apply this approach in systems with limited resources using unsupervised online learning.

Finally, the discriminative keyword spotting, an approach that was introduced by Keshet et al. (2009), should be mentioned. In this approach, instead of using an HMM or similar model, the audio track is embedded in the feature space. Then, a linear (or more complex) model in this space is trained to distinguish *positive* (with a keyword) and *negative* (without a keyword) examples. This allows the use of support vector machine (SVM) approaches to maximize the margin from the separating hyperplane. In addition, the task of training in such a system can be set as the task of maximizing the area under the ROC curve, which is one of the common metrics for assessing the quality of the voice activation system. In such systems, it is necessary to use feature engineering, which can be both an advantage (one can easily embed prior knowledge) and a disadvantage (incorrect prior knowledge leads to poor quality of work; in addition, feature engineering is a complex manual process). In subsequent papers, this approach was developed. Wöllmer et al. (2009b) added

a hidden layer of the bidirectional long short-term memory (LSTM) network as features, Tabibian et al. (2011) used a genetic algorithm instead of a linear classifier, and Tabibian et al. (2014) described the use of a kernel trick within the framework of discriminative keyword spotting. A very detailed explanation of discriminative keyword spotting is offered by Tabibian et al. (2016).

1.4. Training a Keyword Spotter in a Low-Resource Dataset Setup

Chapter 3 concerns the problem of training a highly accurate voice activation system in a low-resource dataset setup. This problem is tackled in several related papers, most often connected to work with low-resourced languages.

Lithuanian and Russian datasets are used to develop new methods for building a keyword spotter in a low-resource setup, i.e., when there are less than 20 training examples per keyword (Chapter 3). This thesis does not focus on the peculiarities of these languages; they are used to demonstrate the application of the methods. Still, the existing research must be mentioned in relation to Lithuanian and Russian.

Generally, there is a lack of papers concerning keyword spotting for under-resourced languages. For example, for the Lithuanian language, there are works about speech recognition (Pipiras et al., 2019; Salimbajevs and Kapociute-Dzikiene, 2018) or speaker identification (Dovydaitis and Rudzionis, 2017; Ivanovas and Navakauskas, 2012), but to the best of author's knowledge no papers focused solely on voice activation. The closest is by Rasymas and Rudzionis (2014), where the authors used several speech recognition models in different languages and a post-classifier to recognize one out of 14 Lithuanian drug names. Unfortunately, this setup uses heavy speech recognizers, so this method is out of the scope of this thesis.

One of the reasons for the small number of papers in Lithuanian is that deep learning methods require relatively big datasets. One of the prominent datasets is LTDIGITS (Rudzionis and Rudzionis, 2002). It contains recordings from 115 male and 225 female speakers, which can not be compared with million-samples datasets used for industrial keyword activation systems.

There is more work concerning Russian keyword spotting and voice processing. E.g., a Russian linguistic processor can be used for word stress definition, homonym disambiguation, and lexicon generation (Smirnov et al., 2016). Mussakhojayeva et al. (2021) improved Russian speech recognition by using a multilingual approach with Kazakh and English languages. Long short-term memory (LSTM) language models can be used for Russian speech recognition (Kipyatkova, 2019).

One of the most productive ideas is to use a bigger out-of-domain dataset in one way or another. For instance, Menon et al. (2019) used an English corpus to

improve the quality of a keyword spotter for an under-resourced Luganda corpus. The authors used a neural network trained as an autoencoder in two scenarios:

- the neural network accepts audio features as input and is trained to output the same audio features;
- the neural network accepts audio features as input and is trained to output the audio features of another audio file but with the same keyword as in the input file.

In both scenarios, the intermediate layers of the neural network have fewer neurons than the number of audio features. This helps to build bottleneck features (the activations of neurons of intermediate levels), which hopefully contain useful information about the audio signal and less noise than the original audio features. In the second scenario, the additional property is achieved: the information about non-keyword traits (such as the gender of the speaker, the speed of pronunciation, etc.) is not necessary for the optimal model (because the model predicts the audio features of another pronunciation of the same keyword). The research showed experimentally that the resulting features are better than the popular Mel frequency cepstral coefficients (MFCC). The neural networks are trained on relatively small datasets (still at least ten times bigger than the setup in Chapter 3). The extracted features are used for the voice activation system which is trained on the datasets with a size comparable to the size of the dataset used in this work. The use of multilingual data is related to the methods of this research, but the proposed setup is a little simpler (the whole model is trained on a multi-dataset, not only feature extractor, which means that the train procedure is the same for both pre-training and training, only datasets are different). Also for some of the proposed methods, there is no need for a relatively big annotated dataset in the target domain.

Knill et al. (2014) presented another example of the use of multilingual data for extracting the bottleneck features. The researchers trained tandem acoustic modeling for phone recognition of several languages and used bottleneck features of this model as acoustic features for speech recognition and keyword spotting for a low-resource target language. This idea was extended by Wang et al. (2015) for pre-training of hybrid speech recognition systems. Tetariy et al. (2015) proposed to create a phone mapping between languages (namely, English and Spanish in their experiments) to transfer a model pre-trained on a big dataset for use on a smaller dataset in another language. Compared to these methods, the methods investigated in Chapter 3 do not require a phone transcription for the corpus used for pre-training and do not require the existence of a phone mapping between languages.

An entirely different way to improve the quality of a keyword spotter in a low-resource setup is to generate synthetic data and use it in training. Lin et al. (2020) used a model pre-trained on 200 million 2-second audio clips to extract bottleneck features. This extractor is used to train a result keyword spotter on a small dataset

augmented with samples generated with speech synthesis. The authors showed that synthetic data is enough to build a high-quality keyword spotter, if the pre-trained feature extractor is good enough. Unfortunately, a good speech synthesis might not be available for some languages.

Another method is to pre-train the model in an unsupervised manner. In such a way, the pre-training dataset can be unlabeled, which hopefully makes it easier to obtain. A very successful way of unsupervised pre-training for automatic speech recognition (ASR) was proposed by Schneider et al. (2019). The authors use a large unlabeled audio corpus to train a model that predicts audio features from the audio features of neighboring frames (similar to a language modeling task). The features extracted from this "language model" are experimentally very useful for speech recognition, especially for very small datasets (1 hour of labeled speech compared to hundreds of hours of a typical speech recognition dataset). Since there were several propositions to use a similar method for voice activation problem with limited datasets (Seo et al., 2021). A disadvantage of this method might be a possibly heavy feature extractor for use in a low-resource consumption setup.

Many methods were proposed to apply deep learning in a low-resource dataset setup for other problems, specifically for speech recognition and image classification. In this work, some of these methods were adapted for a voice activation problem.

Unsupervised pre-training (Erhan et al., 2010) is an example of a general framework, which can be applied when an in-domain labeled dataset is small, but there is a possibility to get a bigger unlabeled dataset. In a unsupervised pre-training scenario, the model is trained on a large corpus of data without target labels (possibly on some artificial problem), and then the model parameters are used to initialize training on the target dataset. Such pre-training can be used in speech recognition (Hinton et al., 2012).

A popular way of pre-training for image classification was proposed by Dosovitskiy et al. (2014). The authors chose several seed images without labels, generated many other samples from the chosen images by using augmentations, and pre-trained the classifier on this dataset. The original seed image and the images generated from it images formed a class for a pre-training classification problem. The classifier trained on this task showed good results on the downstream image processing problem. To the author's best knowledge, the effort (Kolesau and Šešok, 2021b) was the first attempt to adapt this pre-training method to a keyword-spotting task (see Chapter 3).

Semi-supervised learning and, specifically, self-training is another method to improve the quality of a resulting model (Kahn et al., 2020; Triguero et al., 2015) by using unlabeled data. The general framework of self-training is as follows:

1. Train a model on a small labeled dataset.
2. Use the current model to label a bigger unlabeled dataset.

3. Form a new dataset with newly labeled samples by applying the augmentations to input features but leaving the target labels as predicted on the uncorrupted samples.
4. Uptrain the model from the previous step on both the original dataset and the samples from the new dataset.
5. Repeat steps 2–4 until the quality is not improved.

In Chapter 3, the method proposed by Kahn et al. (2020) for speech recognition is applied to a voice activation problem (see Section 3.2.8 for more detail).

1.5. Conclusions of Chapter 1 and Formulation of the Thesis Tasks

The following conclusions can be made for this chapter:

1. The structure of most modern voice activation systems can be described by a template proposed in Section 1.2: acoustic feature extraction pipeline, acoustic model, and the decoding stage. Historically, there have been some papers that did not match this structure (discriminative keyword spotting and DTW were main exceptions), but, for example, all studies on keyword spotters from the Interspeech 2020 conference could be described in that fashion.
2. Most cited works use Mel frequency cepstral coefficients or log Mel-filterbank energy features. In this area, the reduction of the incorporation of the inductive bias into the acoustic feature pipeline must be noted. Recent papers frequently omit discrete cosine transform (DCT) step, probably because deep neural networks work reasonably well with correlated features.
3. In acoustic modeling, widely used GMM are replaced by different kinds of deep neural networks. One of the main research questions in this area is how to apply neural networks in a low-resource setup. Some possible answers are: using specific types of neural network architectures, feature and parameter quantization, and building a cascade model. Another question is how to train deep neural networks (models with a high representation capacity) on low-resource datasets (languages) and to get a good generalization ability.
4. Now, the most voice activation systems use phonemes as acoustic units. Phonemes are stable enough to be reliably found in an audio stream and flexible enough to be used for the majority (if not all) keywords. Another popular choice is to model a whole keyword, especially for works, that evaluate their results on the Google Speech Commands dataset (Warden, 2018).

5. Viterbi algorithm is historically the most popular for a decoding stage, but it is applied mainly for HMM setups.
6. It is a difficult task to compare two voice activation systems in terms of quality. The Google Speech Commands dataset (Warden, 2018) is a step in the right direction. Still, there are too few datasets for a low-resource setup and languages.

Based on the conclusions, the following tasks are formulated to achieve the research goal:

1. To evaluate the possibility of switching to the audio spectrogram feature extraction pipeline due to the simplification trend in acoustic feature pipelines for voice activation systems and investigate hyperparameters of such pipelines.
2. To propose and evaluate new acoustic units for a voice activation system and compare them to existing choices.
3. To propose and evaluate the method to improve the voice activation system repeats detection by modification of the decoding module for phoneme-based voice activation systems.
4. To collect a Lithuanian voice command dataset for the evaluation and further research on voice activation systems for a setup with less than 20 training examples for each keyword.
5. To propose and evaluate deep learning methods for improving detection accuracy for a voice activation system in a setup with less than 20 training examples per keyword.

2

Improving the Quality of Voice Activation by Modifying Parts of a System, Other Than a Neural Network

This chapter presents the research focused on improving the quality of voice activation systems by modifying different parts of such systems, excluding the acoustic model (see 1.2 for the proposed structure of voice activation systems). As was discussed in Chapter 1, most of the research is focused on designing a more accurate neural network, while other important aspects of the systems might be overlooked. Still, this chapter shows that it is possible to significantly improve the quality of keyword detection by fine-tuning the acoustic feature pipeline, carefully choosing the appropriate acoustic unit, or modifying the decoding state of the system.

This chapter starts with an investigation of acoustic feature pipelines for voice activation systems and several hyperparameters of these pipelines. The simplification trend in acoustic feature pipelines is investigated, and the hypothesis of the feasibility of an even further simplification is checked (Section 2.1.4). The Google Speech Commands dataset (Warden, 2018) was used for the experiments. Section 2.1 starts with an experiments' motivation; then, the investigated acoustic feature pipelines are described, and the formal description of the experiment is given. The section finishes with a comparative analysis of acoustic feature pipelines and discusses differences between speech recognition and keyword spotting problems.

Next, the results of using different acoustic units (Section 1.2.3) to build a keyword spotting system are examined. Two new acoustic units were proposed and compared to the known units. These units required less prior linguistic knowledge, which can be helpful in the case of big datasets due to bias-variance tradeoff (Section 2.2.5). While the phoneme option is the best choice in general, the proposed

uniform unit shows better results in the case of large number of training examples. Also, two methods to use acoustic unit examples in training are compared, considering all occurrences and only the occurrences that are part of the target keywords. It was found that the former option gives an 11.78% ± 7.77% (95% confidence interval) better accuracy, than the latter (Section 2.2.6).

Finally, a practical method is proposed to improve the false reject rate (FRR) of a voice-activated system (see Section 2.3.4 for the results). The chapter finishes with a discussion of the obtained results.

The results of the experiments were presented at the international conference and published in the proceedings (Kolesau and Šešok, 2020a, 2021, 2021a).

2.1. Investigation of Acoustic Features for a Voice Activation Problem

This Section examines a simplification trend in acoustic feature pipelines and it proves that the use of prior knowledge in feature extraction is still essential for getting the best results in the keyword spotting problem. Also, it shows that frame length fine-tuning allows to get up to a 1% improvement in detection accuracy for the Google Speech Commands dataset (Warden, 2018) compared to the default values used in open frameworks.

2.1.1. Simplification Trend and Hyperparameters in Acoustic Feature Pipelines

Sound is a continuous physical phenomenon of mechanical vibrations transmission in the form of an acoustic wave. However, most machine learning models do not accept continuous data as input. Thus, the extraction of features from the audio recording has two main goals (see Section 1.2.1 for more detail):

- representing audio in a way suitable for machine learning methods;
- the preservation of the greatest possible amount of information needed to solve the problem (i.e., finding keywords) and the exclusion of the greatest possible amount of information irrelevant to the task ("noise" such as background sounds or the speech variability).

As was discussed in Chapter 1, most modern studies use Mel frequency cepstral coefficients (MFCC) or log Mel-filterbank energy (LFBE) acoustic feature pipelines. Also, there is a trend of simplification of feature pipelines as long as acoustic models (mainly deep neural networks) are getting more powerful and datasets are getting bigger. LFBE feature pipeline is itself simpler than Mel frequency cepstral coefficients (Section 2.1.2), but even simpler options are available. For example, the

audio spectrogram was used by Morgan et al. (1991, 1990) and the raw waveform as used by Kumatani et al. (2017).

Additionally, free parameters in mentioned feature pipelines have not been investigated well for keyword spotting. Two of the most obvious parameters are frame length and frame stride. Bartkova and Jouvet (2015) investigated these parameters in the problem of word alignment by an HMM speech recognition model. Do (2019) used 200 or 400 frames per second for an end-to-end speech recognition model and experimentally proved that these higher rates were beneficial for speech recognition quality. To the best knowledge of the author, there are no similar studies for a keyword spotting problem.

This section examines several acoustic feature pipelines for a keyword spotting model:

- MFCC;
- LFBE;
- audio spectrogram.

2.1.2. Feature Extraction Pipelines Description

Most voice activation systems use a feature extraction approach similar to speech recognition systems (see (Hinton et al., 2012) and Section 1.2.1 for details):

1. The original recording is segmented in possibly overlapping frames.
2. In each frame, a numerical vector that describes the behavior of the sound in this time interval is computed. Usually, this vector is computed using the discrete Fourier transform. Presume this vector has dimension n_{f}.
3. The resulting numerical matrix of the size $T \times n_{\mathrm{f}}$ is used as the result of feature extraction (where T is the number of frames).

MFCC feature pipeline is widely used for voice activation problems (Fernández-Marqués et al., 2018; Manor and Greenberg, 2017; Rose and Paul, 1990; Szöke et al., 2005; Vroomen and Normandin, 1992; Zhu et al., 2013). See Section 1.2.1 for the detailed algorithm of the MFCC computation. An example of MFCC features is presented in Fig. 2.1.

To visualize correlations between different MFCC features, Pearson product-moment correlation coefficients are computed. The Pearson product-moment correlation coefficients $R_{i,j}$ are computed by Equation 2.1:

$$R_{i,j} = \frac{C_{i,j}}{\sqrt{C_{i,i}C_{j,j}}}, \tag{2.1}$$

where $C_{i,j}$ is a covariance of the i-th and j-th features.

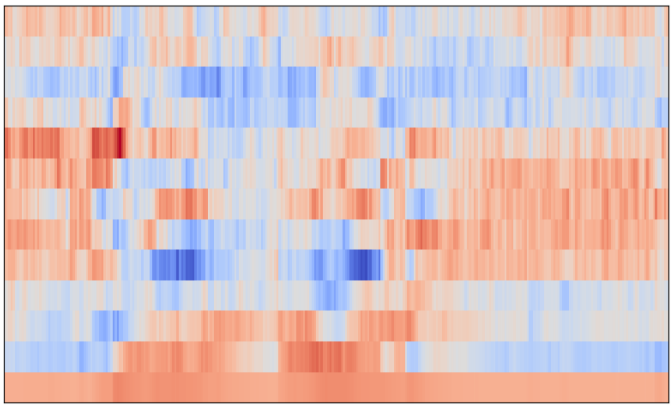


Fig. 2.1. Example of MFCC feature vectors for an audio file

Note that $C_{i,i}$ is (by definition) a variance of i-th feature, which means that $R_{i,i}$ is equal to 1. The values of $R_{i,j}$ are between -1 and 1, inclusively. Essentially, $R_{i,j}$ is a normalized measurement of the covariance, where 1 implies a positive linear correlation (so, x_i grows linearly with growing x_j) and -1 a negative linear one (so, x_i decreases linearly with growing x_j). 0 means a lack of linear correlation (but does not necessarily mean independence). Finally, the matrix R is asymmetric, which means that $R_{i,j} = R_{j,i}$.

The absolute values of $R_{i,j}$ for MFCC features are presented in Fig. 2.2. Note, that the linear correlation is relatively low for all coefficients, which is achieved by a DCT stage of the MFCC pipeline.

Log Mel-filterbank energy (LFBE) is another popular acoustic feature pipeline choice for keyword spotters (Liu et al., 2020; Lopatka and Bocklet, 2020; Wu et al., 2020; Zhang et al., 2020; Zhang and Zhang, 2020). The algorithm for computing these features is the same as computing MFCC but without the last stage (DCT). Note that one of the goals of this step is to decorrelate the features, so omitting this step leads to:

- more correlated features than MFCC;
- small performance gains (less float and complex operations to perform);
- hopefully, more information preserved in the features (DCT is a lossy transformation in float arithmetic).

The example of LFBE features with 26 bins is presented in Fig. 2.3. The adjacent features are much more similar to one another, than for MFCC (Fig. 2.1). The absolute values of Pearson product–moment correlation coefficients for LFBE features are presented in Fig. 2.4 and support the claim of decorrelation properties of the DCT stage.

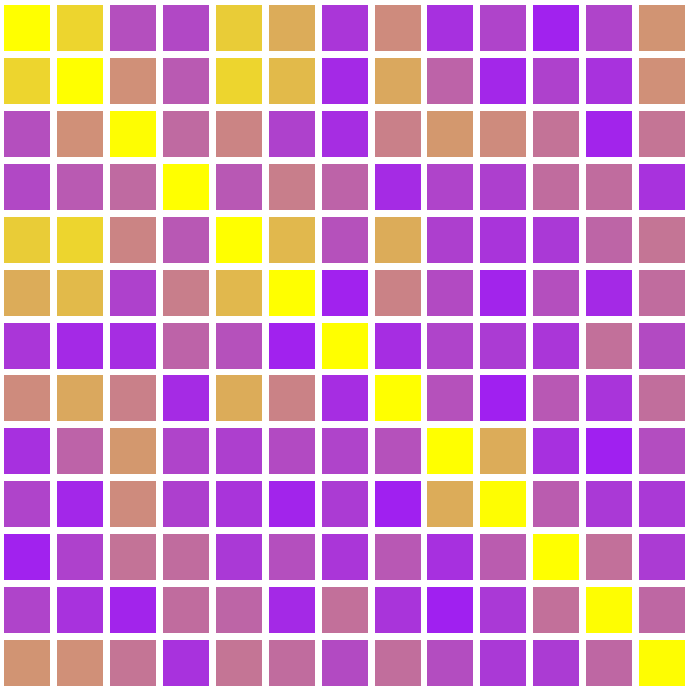


Fig. 2.2. Measure of the linear correlation of MFCC features ($|R_{i,j}|$ from Equation 2.1. Purple means no linear correlation, yellow means a perfect linear correlation

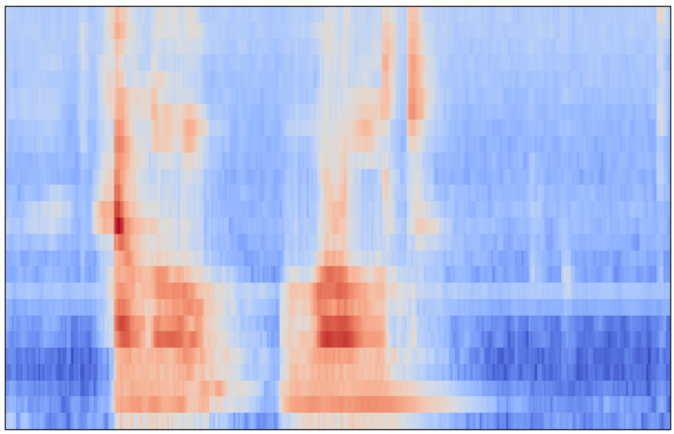


Fig. 2.3. Example of LFBE feature vectors for an audio file

Fig. 2.4. Measure of the linear correlation of LFBE features ($|R_{i,j}|$ from Equation 2.1. Purple means no linear correlation, yellow means a perfect linear correlation

As was discussed in Section 2.1.1, some researchers use a simpler feature pipeline than MFCC or LFBE. The point is that prior knowledge in feature extraction can be more harmful than useful with a powerful enough model and enough training data.

In the experiments, MFCC and LFBE features are compared with an audio spectrogram. Also, it is proposed to apply average pooling to the audio spectrogram to make an input feature more compact.

Computation of audio spectrogram required performing steps 1 and 2 from the MFCC feature extraction pipeline (Section 1.2.1). Note that resulting features are not in a Mel-scale, so a lot of signals are not relevant to actual speech.

Average pooling means that if the number of bins is n_f and the pooling factor is equal to F, then the first F values are replaced with their average, the next F with their average, and so on. Note that the resulting vector will have $\lceil \frac{n_f}{F} \rceil$ features.

An example of an audio spectrogram is presented in Fig. 2.5.

Fig. 2.5. Example of audio spectrogram feature vectors for an audio file

The audio spectrogram after average pooling with factor $F = 10$ is presented in Fig. 2.6.

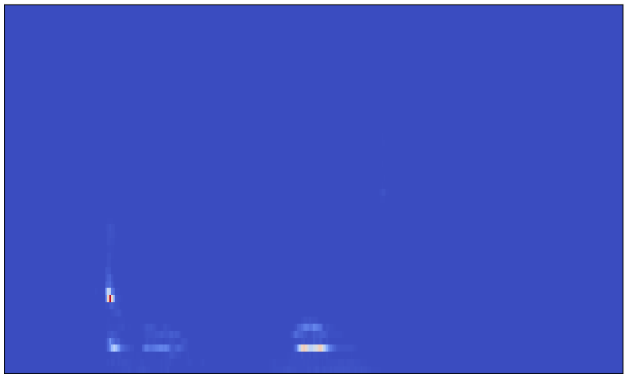


Fig. 2.6. Example of audio spectrogram feature vectors after average pooling with factor $F = 10$ for an audio file

The absolute values of Pearson product–moment correlation coefficients for audio spectrogram features are presented in Fig. 2.7. The numerical value of mean and standard deviations of $|R_{i,j}|$ for all pipelines is presented in Table 2.1. These values are estimated by 1000 audio files from the Google Speech Commands dataset (Warden, 2018). Though standard deviations are rather high for all pipelines, MFCC coefficients are less correlated than spectrogram coefficients, and spectrogram coefficients are less correlated than those from LFBE.

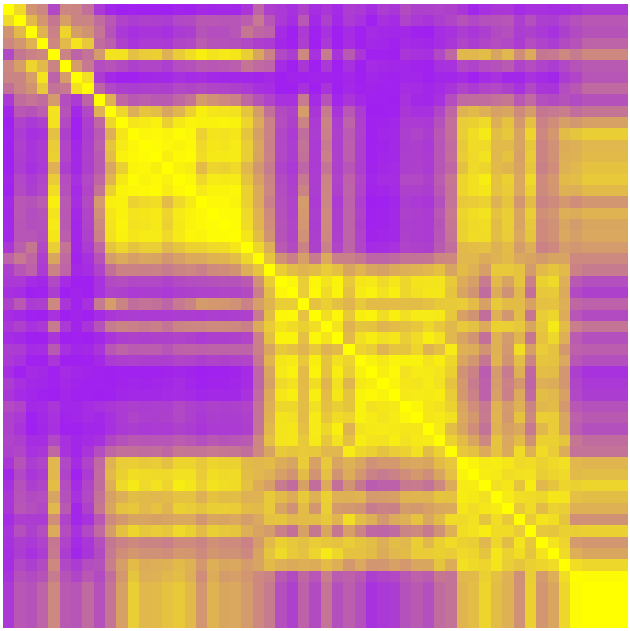


Fig. 2.7. Measure of the linear correlation of audio spectrogram features ($|R_{i,j}|$ from Equation 2.1. Purple means no linear correlation, yellow means a perfect linear correlation

Table 2.1. Mean and standard deviation of $R_{i,j}$-coefficients from Equation 2.1

Pipeline	Measure of the linear correlation
MFCC	0.27 ± 0.21
LFBE	0.76 ± 0.13
Spectrogram	0.45 ± 0.31

2.1.3. Comparing Feature Extraction Pipelines and Hyperparameter Investigation

The experiments used the Google Speech Commands dataset (Warden, 2018). It was released in August 2017 under a Creative Commons license. The dataset contains around 105 829 audio files, 9 981 are used for validation, 11 005 for testing. All audio files are 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 Google implementation (Warden, 2018), the task is to discriminate among 12 classes: “yes,” “no,” “up,” “down,” “left,” “right,” “on,” “off,” “stop,” “go,” unknown and silence.

The experiments used a convolutional neural network similar to cnn-trad-fpool3 (Sainath and Parada, 2015). This model was chosen because it was a popular baseline for small–footprint keyword spotting at the time of the research, i.e., 2018 (Chen et al., 2018; Coucke et al., 2019; Lin et al., 2018). A similar model is still used in TensorFlow "Recognizing Keywords" Tutorial (TensorFlow Core Team, 2021). The model consists of two convolutional layers with stride 1 both in time and frequency. After performing convolution, a pooling layer helps to remove variability in the time-frequency space that exists due to speaking styles, channel distortions, etc. 2×2 non-overlapping max-pooling is used.

Rectified linear unit (ReLU) was chosen as a non-linearity. The dropout rate of 0.5 was used during training. The model consists of 244.2 K parameters.

The model architecture is presented in Fig. 2.8. The convolutional neural network (CNN) takes a 2D "image", in this example $T = 100$, $n_f = 40$. After the first convolutional layer, there are 64 feature maps. Because of the stride of 1, both in time and feature dimensions, all feature maps have the size of 40×100. The filter has a size 8×20. The ReLU non-linearity and a dropout are used. After that, a non-overlapping max-pooling operation is applied with a 2×2 kernel. Because of that, both time and feature dimensions are halved. Then second convolutional layer with a dropout and a ReLU is applied. The final dense layer is used to compute the probability of each class.

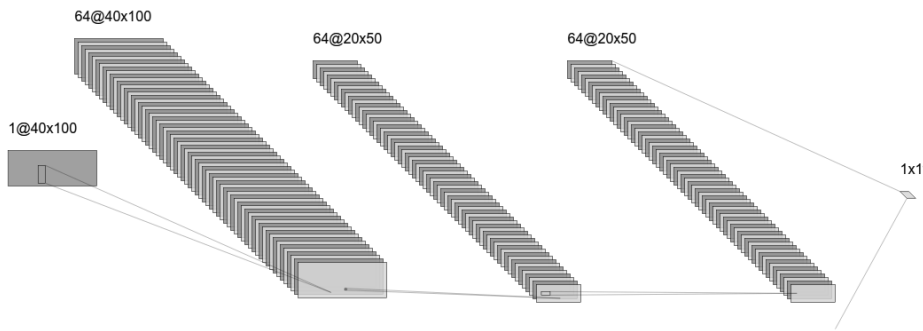


Fig. 2.8. Neural network architecture

To help a neural network learn what sounds to ignore, some audio clips are provided that are neither of the core classes. 10% of training examples from "unknown classes" (which are provided in the Google Speech Commands dataset (Warden, 2018)) are used.

A real voice activation system works in noisy environments. Therefore, it is important to "corrupt" training examples with background noise. "Background

noise" is used from the Google Speech Commands dataset (Warden, 2018), which contains minute-long WAVE files with white noise and recordings of machinery and everyday household activity. Small snippets of these files are chosen at random and mixed at a low volume into clips during training. 80% of the training examples were "corrupted" with background noise with a loudness level of 10%.

A voice activation system needs to activate only if a keyword is uttered. It is safe to assume that the most of the time, no word is uttered. Therefore, 10% of the training examples did not contain any words at all.

The model is trained in two stages:

- 5000 training steps with a learning rate of 0.001;
- 1000 training steps with a learning rate of 0.0001.

Stochastic gradient descent (SGD) was used. Each experiment was repeated three times with different random seeds, and the average accuracy was reported.

2.1.4. Quantitative Results of Acoustic Feature Pipeline Comparison and Hyperparameter Investigation

First, the frame length is set to 30 ms, and the frame stride to 10 ms. These are the default values in the Google Speech Commands tutorial (Warden, 2018). For these fixed parameters, a grid search for the batch size and number of bins (n_f) is run. The results for MFCC are in Table 2.2. The same experiment was conducted for LFBE acoustic feature pipeline (Table 2.3). The results for the audio spectrogram feature pipeline are presented in Table 2.4.

Table 2.2. Grid search results for the MFCC feature pipeline

n_f / batch size	16	32	64	128
20	77.1	79.1	80.0	80.1
40	78.7	79.1	79.7	79.0
60	77.8	78.3	79.0	79.3
80	78.4	78.8	79.9	80.0
100	77.4	78.8	78.2	80.7

Table 2.3. Grid search results for the LFBE feature pipeline

n_f / batch size	16	32	64	128
20	77.0	78.9	79.9	80.1
40	78.5	78.9	80.0	78.9
60	77.4	78.3	78.9	79.3
80	78.4	79.0	79.6	79.9
100	77.1	78.4	78.2	80.7

Table 2.4. Grid search results for the audio spectrogram feature pipeline

n_f / batch size	16	32	64	128
20	48.2	54.6	60.7	61.5
40	51.2	57.9	59.7	60.9
60	47.2	55.0	60.1	63.8
80	48.7	53.7	59.0	63.8
100	52.3	55.4	55.9	67.5

The following points can be suggested from the experiments:

- the choice of MFCC or LFBE feature extraction pipeline is superior to the choice of the audio spectrogram, at least for small datasets. It seems that small CNN can not deduce enough signal from data: but the whole signal is in the data because both MFCC and LFBE are built from the audio spectrogram (see Section 2.1.2 for details). So, prior knowledge in acoustic feature extraction helps for a voice activation problem;
- MFCC is generally better than LFBE, though, for some choices LFBE–based solutions give a better result. Anyway, the difference in quality is about 0.1% on average, which is not very large;
- the number of bins is a robust parameter: even 20 bins is enough to extract the information needed to classify core words;
- it is generally better to use bigger batch sizes.

Next, MFCC feature pipeline is fixed (as the best from the previous set of experiments), and the frame stride and the frame length are investigated. Only the experiments with batch sizes of 64 and 128 are run because of the results of previous experiments. Results for 10 ms stride are summarized in Table 2.5, and for 20 ms in Table 2.6.

The following points can be suggested from experiments:

- a stride of 10 ms is better suited for a voice activation problem; it can be explained by the fact that if a 20 ms stride is used, there are twice fewer frames, so a lot of information about the audio stream is lost. It is not entirely lost because of frame overlapping, but it is not easy for CNN to learn in such a setup. Consequently, frame stacking (Chen et al., 2014a;

Table 2.5. Grid search results for a 10 ms stride

Frame length / batch size	64	128
15	77.7	78.4
20	79.1	79.6
30	78.8	79.2
40	79.9	80.2
55	79.7	79.7

Table 2.6. Grid search results for a 20 ms stride

Frame length / batch size	64	128
15	72.6	73.6
20	74.2	74.8
30	74.7	76.3
40	75.2	76.7
55	76.6	77.3

Fernández-Marqués et al., 2018; Junkawitsch et al., 1997) can be suggested with caution. The accuracy difference is bigger when the frame length is small;

- a greater frame length is generally more advantageous. The downside is that greater frame lengths introduce latency and require more computations. But still, there is a noticeable accuracy boost which can be achieved in a simple way. Notably, many papers (Fernández-Marqués et al., 2018; Manor and Greenberg, 2017; Rose and Paul, 1990; Szöke et al., 2005; Vroomen and Normandin, 1992; Zhu et al., 2013) suggest using a 20–30 ms frame length. An optimized frame length can boost the voice activation system quality (Fig. 2.9).

To statistically support the last statement, confidence intervals were computed for the best hyperparameter choices for each pair of the tested frame length and frame stride. 25 neural network optimizations were performed with random weights initialization. The resulting test accuracies were processed with the bootstrap method from the SciPy library (Virtanen et al., 2020) to compute a 95% confidence interval. The results are presented in Table 2.7. The total number of experiment is equal to 250.

Table 2.7. 95% confidence intervals for test accuracy for the best choices of hyperparameters for a fixed frame length and frame stride

Frame length (ms)	Frame stride (ms)	Mean accuracy	Confidence interval
15	10	78.50	(78.28, 78.72)
20	10	79.53	(79.33, 79.73)
30	10	79.25	(78.97, 79.52)
40	10	80.26	(79.94, 80.58)
55	10	79.81	(79.42, 80.21)
15	20	73.97	(73.70, 74.23)
20	20	74.71	(74.35, 75.06)
30	20	76.41	(76.15, 76.67)
40	20	77.02	(76.76, 77.27)
55	20	77.36	(77.19, 77.53)

Additionally, the following experiment was done. For each frame stride (10 ms and 20 ms), 50 neural network trainings were performed: 25 for one of the baseline frame lengths (20 ms or 30 ms) and 25 for the best frame length from Tables 2.5 and 2.6 for the corresponding frame stride. For each pair of trainings, the gain in accuracy was saved. Then, these 50 gain values were processed with the bootstrap method to a compute confidence interval. The result is that fine-tuning of the frame length in the feature extraction pipeline of a voice activation system allows getting 0.95% ± 0.77% (95% confidence interval) improvement in detection accuracy for the Google Speech Commands dataset (Warden, 2018) and convolutional neural networks compared to the default values of 20 ms and 30 ms. Notably, the standard deviation is somewhat high in this experiment.

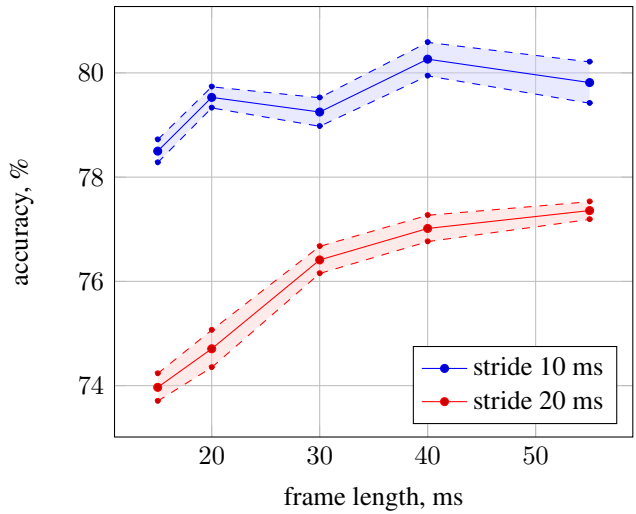


Fig. 2.9. Comparison of a stride of 10 ms and a stride of 20 ms. Dashed lines bound the 95% confidence interval

Finally, the time needed to compute investigated features for a 3-second audio file is measured. Each computation was repeated 1 000 times on Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz. The results are presented in Table 2.8 and the histogram is presented in Fig. 2.10.

Table 2.8. Time consumption for computing acoustic features for a 3-second audio file

Acoustic feature pipeline	Best time	Mean time
MFCC	7.108 ms	14.787 ± 2.6 ms
LFBE	6.099 ms	12.671 ± 4.5 ms
spectrogram	4.134 ms	5.317 ± 0.4 ms

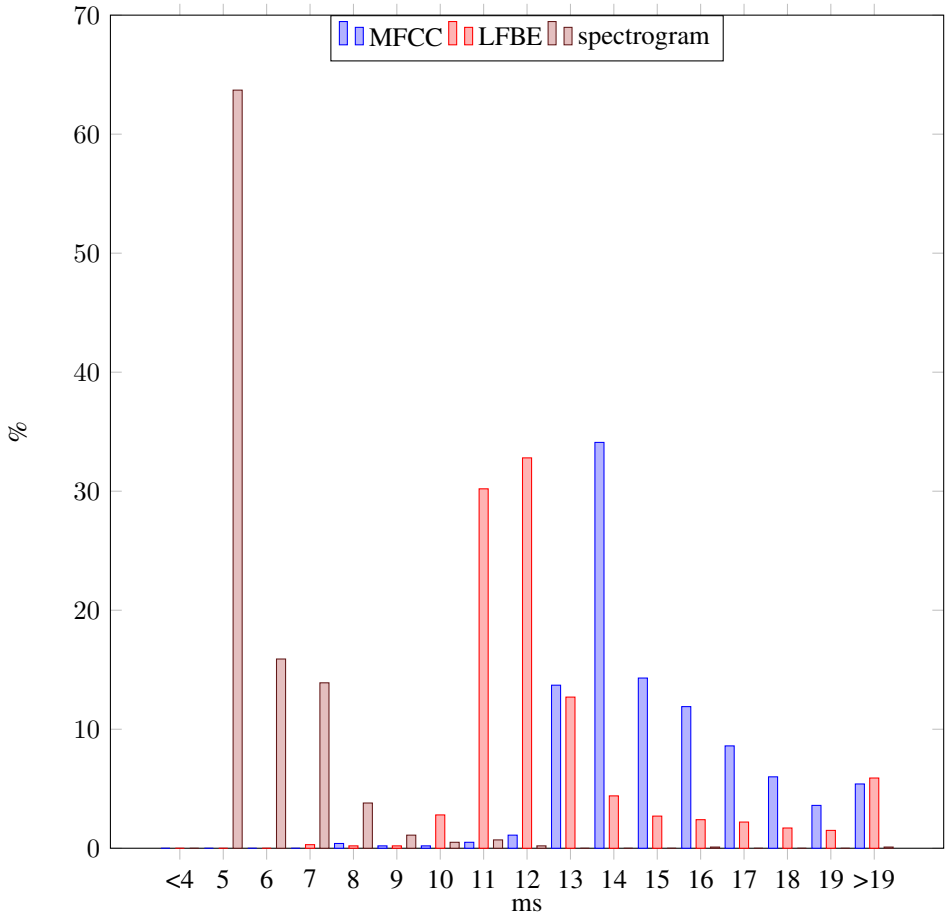


Fig. 2.10. Histogram of the time consumption for computing acoustic features for a 3-second audio file

The results show that there is another tradeoff between MFCC and LFBE features: computation time versus the resulting quality. It needs to be mentioned that an acoustic feature pipeline normally takes much less time than a computing acoustic model (see Section 1.2.2 for detail).

2.1.5. Discussion of Experimental Results for the Acoustic Feature Pipeline Comparison

The experiments showed that prior knowledge in feature extraction is necessary for good results in voice activation. At the same time, several papers show that

it is possible to build a speech recognition system on a raw waveform (Ravanelli and Bengio, 2018; Sainath et al., 2015; Zeghidour et al., 2018). Why is the MFCC pipeline crucial for a simple CNN model and the Google Speech Commands Dataset (Warden, 2018)? The following hypotheses can be suggested:

- the model has too few parameters to learn features from a raw audio spectrogram;
- the Google Speech Commands dataset is too small compared to automatic speech recognition datasets;
- the train regime should be different to learn good features from the raw waveform.

A voice activation system should consume as less CPU as possible, so simplifying the feature extraction pipeline is an important problem.

Also, it was shown that the number of bins, the frame length, and the frame stride are hyperparameters that (though fixed to default in most works) still might be important to achieve the best accuracy. This suggests that a feature extraction pipeline should be considered when building and analyzing state-of-the-art voice activation systems.

2.2. Investigation of Acoustic Units for a Voice Activation Problem

The choice of the acoustic unit greatly influences the quality of the resulting system, given the dataset and the model complexity. Decomposing the keyword into simple acoustic units requires prior knowledge. This knowledge might make the task easier (so the resulting accuracy will be higher), but on the other hand, even slightly incorrect priors could mislead the model, so the quality might drop significantly. The choice of phonemes, syllables, and words are compared, and several synthetic acoustic units for Russian language are proposed. As shown, phonemes are a robust and high-quality choice, especially in a low-resource setting for modern keyword spotting systems.

2.2.1. Motivation of the Keyword Decomposition into Acoustic Units

The requirements of working in real-time and consuming a small amount of CPU make a restriction on the size of the keyword spotting model. This, in turn, limits the complexity and the modeling power of the model. To solve the hard problem of finding the keyphrase, it can be decomposed into detecting simpler acoustic events, such as syllables or phonemes. Prior linguistic knowledge of how a keyword is

decomposed into smaller parts might greatly improve the quality of the system and make it more interpretable. On the other hand, such decompositions are only an approximation of the real world. Building an end-to-end model might be more profitable when a large dataset and a big model are available.

This section examines using the phonemes, syllables, and words as target acoustic events. Also, the uniform and adaptive splittings of the words as acoustic units were proposed, and the quality of resulting voice activation systems was examined. Furthermore, the question of using as targets the events that happen only inside the keyword or all the events of the given type was investigated (e.g., phoneme "g" happens not only in "Google" but also, for example, in "great" – should it be used as a target during the learning of the model?).

As was discussed in Chapter 1, most modern voice activation systems consist of the following parts:

- feature extraction;
- acoustic model;
- decoding.

Feature extraction is the part of converting the source audio stream into acoustic features. Usually, it is done by segmenting the audio and computing an audio feature vector (most usually, Mel frequency cepstral coefficients or log Mel-filterbank energy for each segment (frame) separately). In this paper, 80 log Mel-filterbank energy coefficients are computed for the frames of a 25 ms length and a 10 ms offset.

The acoustic model is a system that generally computes the probability of acoustic observations, which often comes down to computing $P(u|O)$, where u is an acoustic unit and O is acoustic observations). See Section 1.2.3 for a more detailed explanation. In this section, some popular choices for acoustic units are experimentally evaluated, and two new ones are proposed.

The decoding is the process of determining the state sequence with reference to acoustic observation and acoustic model to determine whether a keyword has been uttered or not.

2.2.2. Widely Used Acoustic Units

In this section, some acoustic unit types used in the experiments are described. All units in this section were used before to build voice activation systems.

In phonology, a phoneme is a unit of sound that distinguishes one word from another in a particular language (Wikipedia contributors, 2021). For example, in English, the words "sin" and "sing" are distinguishable by the phoneme "g." The International Phonetic Alphabet is an example of a phonetic alphabet suitable for different languages.

Each phoneme of the target keyword can occur not only in the keyword itself but in other words as well. There is a choice of how to handle these occurrences during the training. The variant where a neural network is optimized to use only the phonemes in the keyword as targets (and handle other cases as fillers) will be called "own phonemes" in this research, and the case where the neural network is optimized to find the phonemes regardless of whether they are a part of the keyword or another word will be called "all phonemes."

"Own phonemes" choice encourages the model to learn the contextual information. It might improve the accuracy of the voice activation system, but at the same time, the real targets are not phonemes anymore, which may "confuse" the model, and the training may fail.

A syllable is a unit for a sequence of speech sounds (Wikipedia contributors, 2021). It is typically made of a syllable nucleus (most often a vowel) with optional initial and final margins (typically consonants). For this research the syllable is considered a sequence of phonemes. As with phonemes, there is the choice of "own syllables" and "all syllables."

The number of syllables per word is almost always less than the number of phonemes per word, so this choice makes a neural network and a decoding a bit simpler, but the duration of a syllable is more variable so the distinction task can be more complex.

As a baseline, the whole word can be used as a target. For a one-word keyphrase there are two neural network outputs: the filler probability and the word itself.

The transition between phonemes can also be used as targets. It reduces the number of targets by one compared to phonemes. The "all" and "own" scenarios are also distinguished for this target type.

2.2.3. Uniform and Adaptive Targets

This section proposed two new acoustic units.

The decomposition of words into phonemes is performed with some assumptions based on linguistic knowledge. Arguably, is has errors just like every model, so it might be profitable to use a simple decomposition and let a neural network deduce actual patterns from data. To check this hypothesis, two methods of decomposing words into sub-word targets were proposed.

First, the word is split into n parts uniformly by the duration of the word. This option is called *uniform targets* in this research. n is chosen via a hyperparameter search.

The phonemes are irregular in duration. Consequently, uniform splitting might result in very different types: several phonemes, one phoneme, and a transition between phonemes. Therefore, the second way of splitting the word is proposed, i.e., *adaptive targets*. In this option, the word is split into n parts, minimizing the sum of

distances from the frame to the average frame in the split in the LFBE space. This can be done effectively via dynamic programming (Section 2.2.4). As for "uniform target," the value of n is chosen via hyperparameter search.

The visual representation of the investigated targets can be seen in Fig. 2.11 for a target keyword with five phonemes.

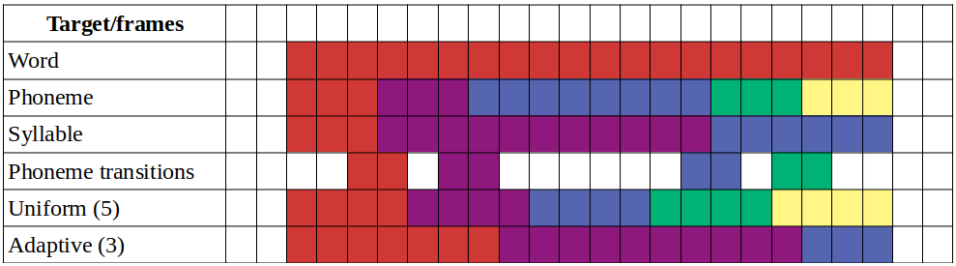


Fig. 2.11. Example of acoustic unit targets

2.2.4. Fast Computation of an "Adaptive" Split

As discussed in Section 2.2.3, the best way must be found for splitting the audio segment into n parts minimizing the sum of distances from the frame to the average frame in the split in the LFBE space. To compute this efficiently, one segment of the split can be considered. Assume $x_1, \ldots, x_m$ are frames of this segment. Then, the "cost" of this segment is:

$$\text{cost} = \sum_{i=1}^{m} \left(x_i - \frac{1}{m} \sum_{j=1}^{m} x_j \right)^2 . \tag{2.2}$$

Equation 2.2 can be expanded:

$$\text{cost} = \sum_{i=1}^{m} \left(x_i^2 - 2x_i \left(\frac{1}{m} \sum_{j=1}^{m} x_j \right) + \left(\frac{1}{m} \sum_{j=1}^{m} x_j \right)^2 \right) . \tag{2.3}$$

By denoting $S = \sum_{i=1}^{m} x_i$, Equation 2.3 can be rewritten as:

$$\text{cost} = \sum_{i=1}^{m} \left(x_i^2 \right) - \frac{2S}{m} \sum_{i=1}^{m} \left(x_i \right) + m \left(\frac{S}{m} \right)^2 . \tag{2.4}$$

Equation 2.4 can be further simplified:

$$\text{cost} = \sum_{i=1}^{m} \left(x_i^2\right) - \frac{2S}{m}S + \frac{S^2}{m} = \sum_{i=1}^{m} \left(x_i^2\right) - \frac{S^2}{m}. \tag{2.5}$$

If one assumes, that $S_i = \sum_{j=1}^{i} x_i$, $S'_i = \sum_{j=1}^{i} x_i^2$, $S_0 = S'_0 = 0$ and uses Equation 2.5, then it is possible to compute the cost of the segment, which starts from the i-th frame and ends with j-th frame, with:

$$\text{cost}_{i,j} = (S'_j - S'_{i-1}) - \frac{(S_j - S_{i-1})^2}{j - i + 1}. \tag{2.6}$$

Algorithm 2.1 summarizes these findings in the procedure that computes costs for all possible frame segments:

Algorithm 2.1 (Computing the costs of all segments)

```
1:  procedure ComputeCosts(x, m)
2:      S ← 0[0 ... m]
3:      S' ← 0[0 ... m]
4:      for i ← 1, m do
5:          S_i = S_{i-1} + x_i
6:          S'_i = S'_{i-1} + x_i^2
7:      end for
8:
9:      cost ← 0[1 ... m, 1 ... m]          ▷ The result, initially a zero matrix
10:     for i ← 1, m do
11:         for j ← i, m do
12:             cost_{i,j} ← (S'_j − S'_{i-1}) − (S_j−S_{i-1})^2 / (j−i+1)   ▷ Equation 2.6
13:         end for
14:     end for
15:
16:     return cost
17: end procedure
```

Algorithm 2.1 has an $\mathcal{O}(m^2)$ complexity.

Algorithm 2.2 is presented as a naive recursive way to find the best "adaptive" split. It recursively generates all possible segment splits. The purpose of some variables of Algorithm 2.2 is presented in Table 2.9.

The procedure returns the best split and its cost.

Table 2.9. Variables used in Algorithm 2.2

Variable	Meaning
n	Target segment number
m	Overall frame number
cost	Segment cost matrix
n'	Current segment number at the time of procedure call
m'	Number of used frames at the time of procedure call
cost$'$	Sum of the costs of the used segments
split	Ending frames of current segments
bsplit	The "best" currently found split
bcost	The cost of the "best" currently found split
ncost	The "new" cost after adding the segment $[m'+1, f]$
nsplit	The "new" split after adding the segment $[m'+1, f]$
csplit	The best split of the "current" recursive branch
ccost	The cost of the best split of the "current" recursive branch

Algorithm 2.2 (Recursive algorithm to find the best split)

```
1:  procedure RecursiveFind(n, m, cost, n', m', cost', split)
2:      if m = m' then
3:          if n ≠ n' then
4:              return ∅, ∞
5:          end if
6:          return split, cost'
7:      end if
8:      bsplit = ∅
9:      bcost = ∞
10:     for f ← m' + 1, m do
11:         ncost ← cost' + cost_{m'+1,f}
12:         nsplit ← split ∪ {f}
13:         csplit, ccost ← RecursiveFind(n, m, cost, n' + 1, f, ncost, nsplit)
14:         if ccost < bcost then
15:             bcost ← ccost
16:             bsplit ← csplit
17:         end if
18:     end for
19:
20:     return bsplit, bcost
21: end procedure
```

Note that the statement inside line 2 of Algorithm 2.2 will be true for all valid splits of m frames into n segments, so $\binom{m}{n}$ times. This means, that Algorithm 2.2 has a $\Omega\left(\binom{m}{n}\right)$ complexity.

Fortunately, it is simple to apply dynamic programming to this problem to make a faster algorithm. Note that if there are two calls to $\mathrm{RecursiveFind}$ procedure with the same values of m' and n', but different values of cost', the one with bigger cost' will not find the resulting solution because if the segments after m' frames are used for both calls, the one with smaller cost' will result with a smaller overall cost.

Introduce the function $d_{n',m'}$ equal to the best cost of n' segments of the first m' frames, and say $s_{n',m'}$ is a corresponding split. It is easy to see, that:

$$d_{n',m'} = \begin{cases} 0, & \text{if } n' = m' = 0; \\ \infty, & \text{if } n' > m'; \\ \infty, & \text{if } n' = 0 \wedge m' > 0; \\ \min\limits_{f=0}^{m'-1} \left(d_{n'-1,f} + \mathrm{cost}_{f+1,m'}\right) & \text{else.} \end{cases} \tag{2.7}$$

Algorithm 2.3 uses Equation 2.7 to compute the best split.

Algorithm 2.3 (Dynamic programming algorithm to find the best split)

```
1:  procedure DynamicProgrammingFind(n, m, cost)
2:      d ← ∞[0...n, 0...m]
3:      d_{0,0} ← 0
4:      s ← ∅[0...n, 0...m]
5:      for m' ← 1, m do
6:          for n' ← 1, n do
7:              for f ← 0, m' − 1 do
8:                  ccost ← d_{n'−1,f} + cost_{f+1,m'}          ▷ Equation 2.7
9:                  if ccost < d_{n',m'} then
10:                     d_{n',m'} ← ccost
11:                     s_{n',m'} ← s_{n'−1,f} ∪ m'
12:                 end if
13:             end for
14:         end for
15:     end for
16:     return d_{n,m}, s_{n,m}
17: end procedure
```

Algorithm 2.3 has an $\mathcal{O}(m^2n)$ complexity, which means, that final Algorithm 2.4 has a complexity of $\mathcal{O}(m^2) + \mathcal{O}(m^2n) = \mathcal{O}(m^2n)$.

Algorithm 2.4 (Compute the best adaptive split)

1: **procedure** BestSplitFind(n, m, x)
2: cost $\leftarrow$ ComputeCosts(x, m) ▷ Algorithm 2.1, $\mathcal{O}(m^2)$
3: **return** DynamicProgrammingFind($n, m,$ cost) ▷ Algorithm 2.3, $\mathcal{O}(m^2n)$
4: **end procedure**

2.2.5. Empirical Evaluation of the Acoustic Unit Choice and Hyperparameters

A Russian private dataset of 174 619 audio files was used for the experiments. 139 516 audio files were used for training, 17 820 for validation, and 17 283 for testing. The phoneme targets were computed via forced alignment with the speech recognition model. The keywords used for the experiments are shown in Table 2.10.

Table 2.10. Keywords used for the experiments

Word	Phonemes	Syllables	Positive samples in test
"Алиса/alisa" (Alice)	5	3	7455
"включи/vklyuchi" (turn on)	6	2	542
"тебя/tebya" (you)	4	2	462
"мне/mne" (me)	3	1	889

A time-delay neural network (TDNN), introduced by Zeppenfeld and Waibel (1992), with ReLU as a non-linearity, was used. This model was chosen because, at the time of the research, it was shown to be more effective than the convolutional models used in Section 2.1 (Myer and Tomar, 2018). The layer specifications are presented in Table 2.11. The whole receptive field at frame f is made up of the frame $f-28$ to the frame $f+12$. The number of units in the last layer is determined by the chosen targets. For example, if the model trained to distinguish phonemes of the word "Alice," it needed six outputs: one for the filler and five for the phonemes.

All the models are trained with an SGD optimizer for per-frame cross-entropy loss. The learning rate and batch size were chosen via random search. The training is performed for ten epochs. If the validation loss between consecutive epochs differs by less than 0.001, the learning rate is decayed by the factor of $\frac{4}{3}$.

Table 2.11. Specifications of the used neural network

Layer	Units	Context
1	192	[−8, 1, 2]
2	96	[−3, 2]
3	192	[−8, 1, 2]
4	96	[−3, 2]
5	192	[−3, 2]
6	96	[−3, 2]
7	-	[−0]

2.2.6. Results and Discussion of the Empirical Evaluation of the Acoustic Unit Choice

25 runs were made for each keyword and each target. The threshold was tuned in such a way that the false reject rate (FRR) on the test set was less than 0.1, and the false alarm rate (FAR) was minimized. The numbers of positive samples were very different for different words, so the relative drop in the quality compared to the best option for the given word is reported. The results are in Table 2.12. The best options are highlighted in bold. The value n is specified in the brackets when appropriate.

Table 2.12. Increase in the best false alarm rate compared to the best choice

Target / Keyword	Алиса (alisa)	Включи (vk-lyuchi)	Тебя (tebya)	Мне (mne)
own phonemes	5%	1%	N/A	54%
all phonemes	36%	**0%**	**0%**	**0%**
own syllables	17%	150%	114%	141%
all syllables	34%	131%	105%	137%
own phoneme transitions	49%	24%	123%	88%
all phoneme transitions	22%	53%	91%	102%
uniform target	**0% (6)**	19% (6)	95% (3)	35% (7)
adaptive target	10% (5)	28% (8)	152% (2)	54% (7)
word target	80%	733%	176%	139%

The experiments show that phonemes could be used as a baseline for a keyword spotting system. On average, it shows the best accuracy (Table 2.13) and wins in three settings out of four. At the same time, the clear second option is the proposed uniform target, which shows the highest accuracy in the case of the biggest number of keyword utterances in the training set. It can be explained with an inductive bias framework: when the dataset is smaller more prior knowledge needs to be incorporated into the learning task, which makes it easier. In this research, the linguistic assumption is used that the keyword consists of the sequence of the

phonemes, and this sequence is the same for all keyword pronunciations. Moreover, a forced alignment tool is used to find the audio segments corresponding to the phonemes, which may output slightly incorrect results. This means that at some point, the incorporated bias may even harm the learning, so less restrictive target units should be used. Another advantage of the proposed uniform target is that there is no need for a forced alignment tool to find phoneme borders. However, the keyword boundaries are must still be known.

The proposed adaptive target shows the results comparable to the phoneme transition option but similarly does not require the linguistic assumptions. Still, the experiments provide no reason to use either of these options over the uniform target or phonemes. In the case of an adaptive target, it can be hypothesized that the Euclidean distance in the LFBE space is not very indicative of the audio change (as per the human perspective) because it "penalizes" the change in all Mel-spectra equally, which is not physically justified, to the best of the author's knowledge.

Table 2.13. Average increase of the best false alarm rate compared to the best choice

Target / Keyword	Алиса (alisa)	Включи (vklyuchi)	Тебя (tebya)	Мне (mne)	Overall
phonemes	5%	**0%**	**0%**	**0%**	**1.25%**
syllables	17%	131%	105%	137%	97.5%
phoneme transitions	22%	24%	91%	88%	56%
uniform target	**0% (6)**	19% (6)	95% (3)	35% (7)	37.25%
adaptive target	10% (5)	28% (8)	152% (2)	54% (7)	56%
word target	80%	733%	176%	139%	282%

Finally, the choice between using all the occurrences of the acoustic units in the training set as the learning targets ("all" option) and using only the acoustic units that are uttered inside the target keyword ("own" option) is investigated. Based on Table 2.14, the "own" option is preferable when the dataset is large or the target word is rather long. In these cases, the model can take advantage of the contextual information. When a dataset is small, choosing "own" options greatly reduces the number of positive samples for each target, so the optimization becomes hard. Because of the same reason, the "word" target does not show good results, because most of the examples are negative. On average, the "all" option is preferable.

Table 2.14. Average increase of the best false alarm rate compared to the best choice

Option / Keyword	Алиса (alisa)	Включи (vklyuchi)	Тебя (tebya)	Мне (mne)	Overall
own	23%	58%	118%	94%	73%
all	31%	61%	65%	80%	60%

To statistically support the last statement, the confidence intervals were computed as follows. For each keyword, 25 experiments were carried out. Each experiment consisted of a random choice of the acoustic unit and training of two neural networks: one for the "all" option and one for the "own" option. In 58 cases, "all" option was better, in 42 cases, the "own" option was better. The resulting differences in the test accuracy were processed with the bootstrap method from the SciPy library (Virtanen et al., 2020) to compute a 95% confidence interval. The resulting mean difference was 11.78% with a 95% confidence interval (4.01%, 19.54%).

2.3. Detecting Keyword Repeats in Voice Activation Systems

In this section, a practical method is proposed to improve the false reject rate (FRR) of a voice-activated system. Users of such systems tend to repeat the activation word in the case of false rejection.

The proposed method to modify a decoder (see Section 1.2.4 for details) of a phoneme-based voice-activated system for detecting such repeats allows catching 14.6% of previously undetected cases while not increasing the false alarm rate and 79% while adding less than 1% of the false alarm rate. The fact that the repeat of the activation word consists of twice as many phonemes than the single pronunciation allows more reliable detection. At the same time, these phonemes are the same as in the original keyword, so there is no need to retrain the acoustic model.

The section starts with the motivation of the decoding stage modification, then the formal description of the experiment is given.

2.3.1. Improving the False Reject Rate by Detecting Keyword Repeats

Most modern voice-activated systems use machine learning models for keyword detection (see Chapter 1 for more details). This section proposed the method to improve the quality of the keyword detection (keyword spotting), i.e., the detection of the pre-defined keyword or keyphrase in the audio stream.

The voice-activated system has two types of mistakes, concerning the keyword detection:

- false activation (detection of the keyword when it wasn't uttered);
- false rejection (skipping the detection of the uttered keyword).

Therefore, the quality of voice-activated systems is often measured by the false alarm rate (FAR, the rate of false detection on the sample without keyword) and the false reject rate (FRR, the rate of false rejection on the sample with the keyword).

See Section 1.2.5 for the detailed explanations of these and other metrics of a voice activation system.

The proposed method aims to improve the FRR without compromising the FAR in some practical scenarios. Specifically, the decoder of a phoneme-based keyword spotter, similar to the one that can be used for "Ok, Google" activation in Android mobile devices (Chen et al., 2014a), is modified. In this setup, the additional computations are negligible.

It can be noticed that the users, at least of some voice-activated systems, tend to repeat the keyword in case of false rejection (instead of, for example, using a manual control). If the design of the system allows, it is proposed to modify the decoding stage of voice activation by additionally detecting such repeats. It can be argued that such repeats are easier to detect because they consist of twice as many acoustic events (e.g., phonemes) than the single keyword pronunciation.

In Section 1.2, most voice-activated systems can be described as consisting of:

- feature extraction;
- acoustic model;
- decoding stage.

The first stage is about converting the source analog signal into features that can be used by machine learning models. Most systems perform audio segmenting into the frames (with a typical duration of 20–30 ms) and compute the spectrogram or its transformation for each frame. See Section 2.1 for a detailed explanation.

An acoustic model is a part of a voice activation system that computes the probability of acoustic observations, which often comes down to computing $P(u|S)$, where u is an acoustic unit and S are acoustic observations, as was discussed in Section 1.2.2. There are several popular choices for the acoustic units: graphemes, phonemes, and syllables (Section 1.2.3). The method proposed in this section works for all these options, but the experiments are carried out for a phoneme-based voice activation system. An acoustic model is often represented by a neural network (as in this section).

The decoding stage (or decoder) is used to decide detection based on the output of an acoustic model. It might vary from simply comparing the keyword probability with the threshold to performing the forward-backward algorithm on the decoding lattice (Section 1.2.4). In this section, the decoding method proposed by Chen et al. (2014a) is modified. A detailed explanation of the method is provided in Section 2.3.2.

2.3.2. Proposed Modification for the Keyword Repeats Detection

As in paper by Chen et al. (2014a), a neural network is used as an acoustic model, LFBE as a feature extraction pipeline, and the aggregated acoustic model outputs are compared to the threshold value, which controls the balance between FAR and FRR. In the first stage (feature extraction), log Mel-filterbank energy are computed for the source audio. Next, the neural network is applied. Each output gives the probability of the specific acoustic event (a phoneme, in the discussed case) happening in the current frame. Finally, all these probabilities are aggregated to get the resulting probability of the keyword utterance. See Fig. 1.1 for the illustration.

The acoustic model output aggregation (i.e., the decoding) is performed in two steps (following Chen et al. (2014a)). Assume that the output of the acoustic model is the 2-dimensional tensor: time frames by neural network outputs (the probability of the given acoustic event at the given timestamp). Firstly, the noisy neural network outputs are smoothed by averaging on the sliding window (the continuous segment) of w_s frames. Denote by $p_{f,i}$ the i-th (out of n_i) output of the neural network at the frame f (out of n_f). Then, the smoothed value $p'_{f,i}$ (probability of the given event smoothed over time window) is computed by:

$$p'_{f,i} = \frac{1}{f - H + 1} \sum_{k=H}^{f} p_{k,i}, \tag{2.8}$$

where $H = \max\{1, f - w_s + 1\}$ is the index of the first frame within the sliding window.

The straightforward implementation of the smoothing is presented in Algorithm 2.5.

Algorithm 2.5 (Probability smoothing: naive version)

procedure SmoothProbabilitiesNaive(p, w_s, n_f, n_i)
    $r \leftarrow 0[1 \ldots n_f, 1 \ldots n_i]$ ▷ The result, initially a zero matrix

    for $f \leftarrow 1, n_f$ **do**
        $H \leftarrow \max\{1, f - w_s + 1\}$
        for $i \leftarrow 1, n_i$ **do**
            for $j \leftarrow H, f$ **do**
                $r_{f,i} \leftarrow r_{f,i} + p_{j,i}$
            end for
            $r_{f,i} \leftarrow \dfrac{r_{f,i}}{f - H + 1}$
        end for

```
12:     end for
13:
14:     return r
15: end procedure
```

The for-loop in lines 7–9 of Algorithm 2.5 has a $\mathcal{O}(w_s)$ complexity. The for-loop in lines 6–11 performs n_i iterations, while the for-loop in lines 4–12 performs n_f iterations. Therefore, the whole Algorithm 2.5 has a complexity of $\mathcal{O}(n_f n_i w_s)$.

It is possible to obtain a faster procedure by changing the summation order. The faster smoothing algorithm is presented in Algorithm 2.6.

Algorithm 2.6 (Probability smoothing: faster version)

```
1: procedure SmoothProbabilities(p, w_s, n_f, n_i)
2:     r ← 0[1 ... n_f, 1 ... n_i]                ▷ The result, initially a zero matrix
3:
4:     for i ← 1, n_i do
5:         s ← 0      ▷ Variable, to aggregate the numerator sum of Equation 2.8
6:         for f ← 1, n_f do
7:             s ← s + p_{i,f}
8:
9:             if f < w_s then          ▷ Compute the denominator of Equation 2.8
10:                d ← f
11:            else
12:                d ← w_s
13:            end if
14:
15:            r_{i,f} = s / d
16:
17:            if f ⩾ w_s then
18:                s ← s − p_{i,f−w_s+1}
19:            end if
20:        end for
21:    end for
22:
23:    return r
24: end procedure
```

In this version, a variable s is used to contain the current value of the numerator from Equation 2.8. In line 7 of Algorithm 2.6, the value $p_{i,f}$ is added to s, and after

w_s iterations, it is subtracted in line 18 because by that time it is gone from the summation window. The body of the for-loop in lines 6–20 has a complexity of $\mathcal{O}(1)$ and is executed for n_f iterations. The for-loop in lines 4–21 is executed for n_i iterations. This means that the overall complexity of the procedure in Algorithm 2.6 is $\mathcal{O}(n_i n_f)$, which is better than the complexity of Algorithm 2.5.

The probability P_f of the keyword ending in the frame f is computed by the following equation within the sliding windows of $w_{\max}$ frames.

$$P_f = \max_{F \leqslant f_1 < f_2 < \cdots < f_n = f} \sqrt[n]{\prod_{i=1}^{n} p'_{f_i,i}}, \tag{2.9}$$

where $F = \max\{1, f - w_{\max} + 1\}$ is the index of the first frame within the sliding window, f_i is the index of the frame where the i-th acoustic event has happened according to the model, n is the number of consecutive acoustic events that must happen for the whole keyphrase to happen. For example, for the keyword "Alice" to happen, four following phoneme events must happen: /æ/, /l/, /i/, and /s/.

A more general method of aggregation from Equation 2.9 is presented in Algorithm 2.7. The version assumes that some acoustic units may occur in the keyword sequence several times. This sequence is given to the algorithm by a vector a of the length n.

Algorithm 2.7 (Probability aggregation)

```
1:  procedure AgreggateProbabilities(p', w_max, n_f, n_i, a, n)
2:      r ← 0[1 ... n_f]                          ▷ The result, initially a zero vector
3:
4:      if n > w_max then                          ▷ No keyword sequence is possible
5:          return r                               ▷ That's why all the probabilities are zero
6:      end if
7:
8:      for f ← n, n_f do          ▷ Note, that P_f = 0 for f < n by the definition
9:          d ← 0[0 ... n − 1] ▷ d_i is the maximal probability product of the first i
    elements of the sequence
10:         d_0 = 1                                ▷ The helper value
11:
12:         F ← max{1, f − w_max + 1}
13:         for j ← F, f − 1 do
14:             for i ← n − 1, 1 do                ▷ Note the reverse ordering
15:                 d_i ← max{d_i, d_{i−1} · p'_{j,a_i}}
16:             end for
17:         end for
```

18:
19: $r_f \leftarrow \sqrt[n]{d_{n-1} \cdot p'_{f,a_n}}$
20: **end for**
21:
22: **return** r
23: **end procedure**

There are $\binom{w_{\max}}{n}$ ways to choose n indexes from $w_{\max}$ array in Equation 2.9. Note that because of that, a naive implementation of Equation 2.9 would have the complexity $\mathcal{O}(n_f \cdot \binom{w_{\max}}{n} \cdot n)$, which is prohibitively slow for real use. Consider the practical case, where the frame stride is 10 ms, the $w_{\max} = 100$ (so, the keyword is believed to have a duration not more than one second), and $n = 5$ (the keyword has five phonemes, like "Алиса" (alisa) in this section). In such a case, for each value of f, one needs to consider $\binom{w_{\max}}{n} = \binom{100}{5} = 75\,287\,520$ variants to choose f_i (Equation 2.9). For one second of the audio, such an operation needs to be performed 100 times (because the frame stride is 10 ms).

To speed up the computations, the dynamic programming is used in Algorithm 2.7. For simplicity, fix the value of $f = w_{\max}$, so the first available frame is 1, and the last is $w_{\max}$. For all other values of f, the according index shift can be performed. Consider the function $d_{j,i}$:

$$d_{j,i} = \max_{1 \leqslant f_1 < \ldots < f_i \leqslant j} \prod_{k=1}^{i} p'_{f_i,a_i}, \tag{2.10}$$

where $d_{j,i}$ equals to a maximal product of probabilities of the first i elements of the keyword sequence, where all indexes are in the order and in the first j frames of the audio.

It is easy to see, that if one cares only about the maximal value of P_f for the whole audio, all that is required to do is to consider $d_{w_{\max},n}$ only. The formal difference is that $d_{w_{\max},n}$ may be achieved in the case when $f_n \neq f$, but this only means, that $P_g > P_f$ for some $g < f$. If the real value of P_f is actually needed, the following equation can be used:

$$P_f = \sqrt[n]{d_{f-1,n-1} \cdot p'_{f,a_n}}. \tag{2.11}$$

Algorithm 2.7 uses the fact, that there is no need to store the whole d matrix, only the last row is needed. Consequently, only a vector d is defined in line 9. Note the dimensions of that vector: the helper value of $d_0 = 1$ is used, so there is no need in out-of-boundaries check in line 15. At the same time, there is no d_n value because it is only needed at the last step (Equation 2.11). Because of the reverse

ordering in the for-loop (line 13) all the values of d_k for $k \leqslant i$ are related to the frame $j-1$, while all the d_k for $k > i$ are related to the frame j. Equation 2.11 is used in the line 19 to compute the final value.

In the actual implementation of Algorithm 2.7 there are several practical differences:

- all the probabilities arithmetic is performed in the logarithmic domain, so there are less floating-point arithmetic errors;
- the root extraction in line 19 is omitted, because this function is monotonic and it is computationally easier to raise the threshold value to the n-th power.

Finally, it should be noted, that the overall complexity of Algorithm 2.7 is $\mathcal{O}(n_f \cdot w_{\max} \cdot n)$.

The whole procedure of decoding is presented in Algorithm 2.8. The procedure returns a Boolean value "was there a keyword in the audio stream." The overall complexity of the algorithm is $\mathcal{O}(n_f n_i) + \mathcal{O}(n_f w_{\max} n) = \mathcal{O}(n_f(n_i + w_{\max} n))$.

Algorithm 2.8 (Decoding algorithm)

1: **procedure** Decoding($p, w_s, w_{\max}, n_f, n_i, a, n$, threshold)
2: $p' \leftarrow$ SmoothProbabilities(p, w_s, n_f, n_i) ▷ Algorithm 2.6
3: $P \leftarrow$ AgreggateProbabilities($p', w_{\max}, n_f, n_i, a, n$) ▷ Algorithm 2.7
4: **return** $\max P \geqslant$ threshold
5: **end procedure**

The activity diagram in unified modeling language (UML) for processing one probability frame is presented in Fig. 2.12. Unlike Algorithm 2.8, UML shows how the algorithm can be used in a streaming fashion (where audio is not fully known before the start of the algorithm).

The proposed method works for any acoustic model that computes the probability matrix p. For the experiments, a pre-trained time-delay neural network (TDNN) is used, introduced by Zeppenfeld and Waibel (1992), with a rectified linear unit (ReLU) non-linearity. The offsets of the layers are presented in Table 2.15.

2.3.3. Proposed Method for the Keyword Repeats Detection

The keyword "Алиса" (alisa) is used for the experiments. It consists of five phonemes: /a/, /ll/ (soft /l/), /i/, /s/, /schwa/. To model the repeats, it is proposed to add another target by repeating the target phoneme sequence. Equation 2.9 transforms to computing the probability of 10-phoneme sequence: "a-ll-i-s-schwa-a-ll-i-s-schwa." This takes additional computation time, but in practice, this change is vastly dominated by the time needed for the acoustic model inference.

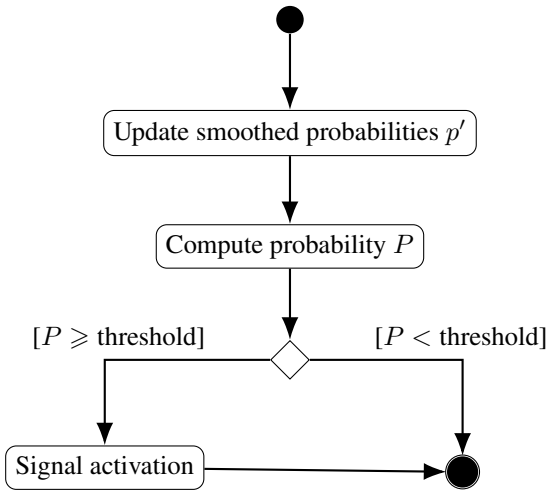


Fig. 2.12. Activity UML-diagram of processing one probability frame by the investigated decoding module

Table 2.15. Specifications of the Used Neural Network

Layer	Units	Context
1	192	[−8, 1, 2]
2	96	[−3, 2]
3	192	[−8, 1, 2]
4	96	[−3, 2]
5	192	[−3, 2]
6	96	[−3, 2]
7	-	[0]

It can also be argued that the detection of such a repeat target is more reliable than the detection of a single utterance because twice as many acoustic events needed to happen for the detection to occur. It is supported by the results in Section 2.3.4.

It is proposed to use both the single and repeat targets in the voice-activated system. The system is activated if at least one of the targets signals the detection. This way, the system recall is improved (i.e., a decrease in the FRR) by detecting the repeats that are not detected by the baseline model. On the other hand, new false alarms might be introduced, but hopefully, not many because the false alarm of the new target is much less probable than the false alarm of the original target. Denote the original threshold as t, the repeats threshold as t', the original $w_{\max}$ as $w_{\max}$, and the repeats $w_{\max}$ as $w'_{\max}$. The whole procedure is presented in Algorithm 2.9:

The activity diagram in a unified modeling language for processing one probability frame with the proposed modification is presented in Fig. 2.13. Unlike

Algorithm 2.9, UML shows how the algorithm can be used in a streaming fashion (where audio is not fully known before the start of the algorithm). It also highlights that the smoothed probabilities need to be computed only once.

Algorithm 2.9 (Modified decoding algorithm)

procedure Decoding($p, w_s, w_{\max}, w'_{\max} n_f, n_i, a, n, t, t'$)
  $p' \leftarrow$ SmoothProbabilities(p, w_s, n_f, n_i) ▷ Algorithm 2.6

  $P \leftarrow$ AgreggateProbabilities($p', w_{\max}, n_f, n_i, a, n$) ▷ Algorithm 2.7
  if $\max P \geqslant t$ **then**
    return true
  end if

  $a \leftarrow$ Concatenate(a, a)
  $P \leftarrow$ AgreggateProbabilities($p', w'_{\max}, n_f, n_i, a, 2n$) ▷ Algorithm 2.7
  return $\max P \geqslant t'$
end procedure

To quantify the benefits of the method, only the following samples were left in datasets:

- samples with repeats of the target word, all skipped by the baseline model;
- samples without keywords and the baseline model inactivated.

The first part of the resulting dataset is where the new target can detect previously undetected keywords. The hit-rate is reported on this part (the fraction of the utterances with a detection). The second part consists of the sample in which a false alarm could happen, additionally to the baseline model. This leaves around 28 000 samples for both the development and hold-out set.

2.3.4. Empirical Results of the Proposed Method

For the experiments, a private dataset with utterances in Russian was used. The keyword "Алиса" (alisa) was chosen. The whole dataset consists of 90 000 samples with an average duration of 6 seconds, where around 40% contain at least one keyword pronunciation. The dataset was split into two equal parts: one for the hyperparameter tuning and the hold-out set for the metrics computation.

One of the advantages of the proposed method is that there's no need to retrain the acoustic model; the only change is in the decoding. Nevertheless, the description of the training process is provided for getting the baseline acoustic model. This allows reproducing the findings of the thesis.

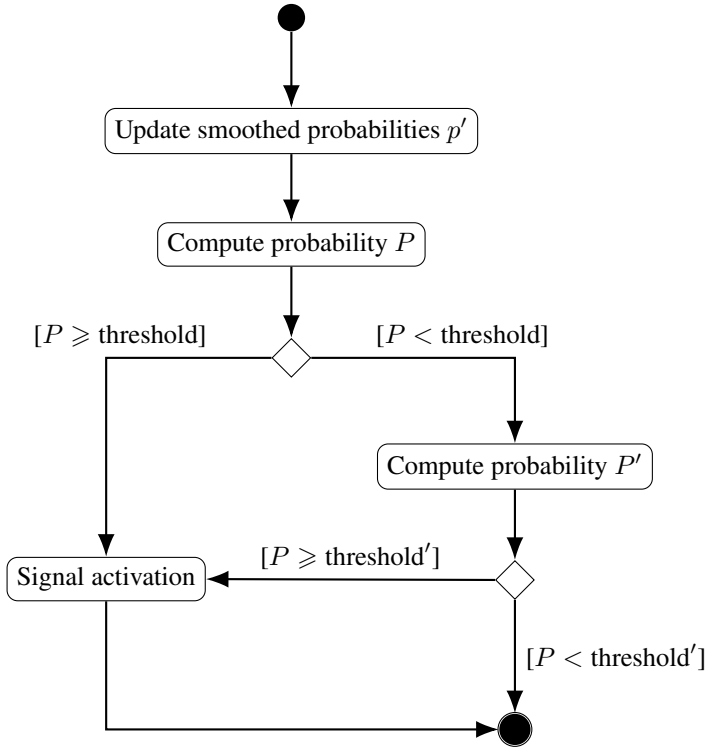


Fig. 2.13. Activity UML-diagram of processing one probability frame by the proposed modification of the decoding module

A Russian private dataset was used. This dataset contains around 400 000 utterances by 100 different people, which was randomly split into three parts: trainset (80 speakers), the validation set (10 speakers), and the testset (10 speakers). These utterances were recorded on mobile devices. The dataset lacks background noise samples, so samples from the Google Speech Commands dataset (Warden, 2018) were reused. Each training sample was augmented with noise with a 10% probability.

The neural network was trained with per-frame cross-entropy loss for classification on six targets: five phonemes and one silence class. The targets for each utterance were computed by force aligning the audio with the speech recognition model. The neural network was trained for five epochs with the SGD optimizer. The learning rate was chosen via a hyperparameter choice, and was multiplied by 0.5 at the end of each epoch.

There is no need to retrain an acoustic model, which is one of the advantages of the proposed method. Therefore, there are only a few parameters that can be tuned

to achieve better results. A grid search was performed for the activation threshold for different values of $w_{\max}$. It is important to increase the value of $w_{\max}$, because the repeat of the activation word has a length at least twice bigger than the single activation word. The effect of leaving the $w_{\max}$ the same can be seen in Table 2.16. It is worth noting that increasing the $w_{\max}$ past 300 has no effect on the hit rate but introduces more computations.

Table 2.16. Hit rate without adding false alarms

$w_{\max}$	Hit rate
100	7.69%
200	13.84%
300	14.61%
400	14.61%

The result of the threshold grid search for the values that yield a false alarm rate (FAR) less than 1% (reasonable operation points for practical use) is presented in Fig. 2.14. Different values of $w_{\max}$ give very close results except the point with a very low false alarm rate. The value $w_{\max} = 300$ can be chosen for a good trade-off between the accuracy and the amount of computation.

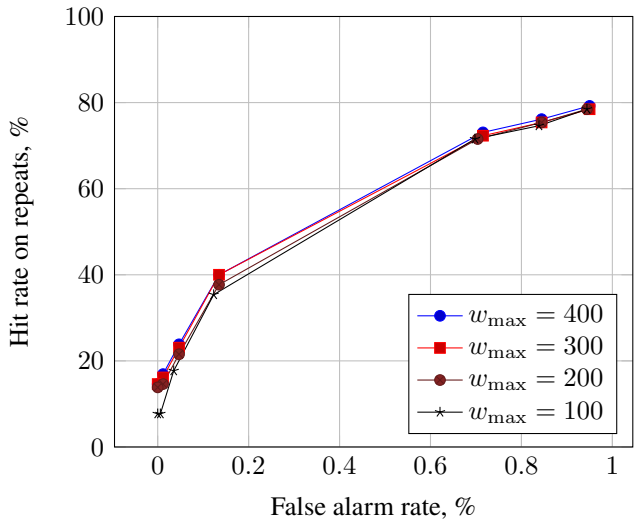


Fig. 2.14. Quality of repeat detection on operation points with the false alarm rate of less than 1%

The full results of the grid search for $w_{\max} = 400$ are presented in Fig. 2.15. The hit-rate increase quickly saturates after 1% of FAR. In the experiments, the

model gets 5.1% FAR for detecting 99% of keyword repeats that are not detected by the original model.

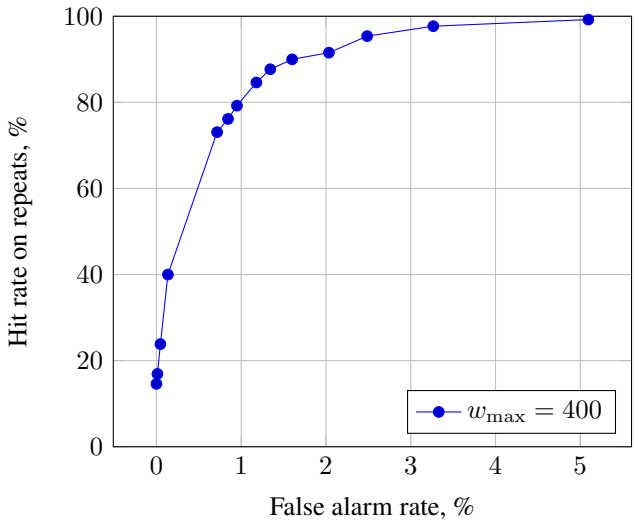


Fig. 2.15. Quality of repeat detection on all operation points

To statistically support the claims of this section, confidence intervals were computed. 25 neural network optimizations were performed with random weights initialization. Then, the proposed modification was applied, and the detection rate of previously undetected repeats was computed for two cases: not allowing any new false alarms and allowing 1% new false alarms. The resulting repeat hit rates were processed with the bootstrap method from the SciPy library (Virtanen et al., 2020) to compute 95% confidence interval. The results are presented in Table 2.17.

Table 2.17. Confidence intervals for the repeats hit rate

FAR increase	Mean hit rate	Confidence interval
0%	13.31%	(11.57%, 15.08%)
1%	79.87%	(78.37%, 81.38%)

To measure resource consumption of the method, it is useful to compare the baseline (i.e., the original model without repeat detection) and the modified model with different values of $w_{\max}$. The models were inferred on a one hour long audio file using one core of Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz 25 times, and the RAM use and process time were measured. The results are presented in Table 2.18.

Table 2.18. Resource consumption

Model	RAM (MiB)	Process time (s)
baseline	8.14 ± 0.11	76.96 ± 0.97
$w_{\max} = 100$	8.08 ± 0.10	77.95 ± 1.18
$w_{\max} = 200$	8.13 ± 0.11	78.66 ± 0.64
$w_{\max} = 300$	8.16 ± 0.08	79.26 ± 1.37
$w_{\max} = 400$	8.07 ± 0.11	79.87 ± 1.27

The RAM use is actually the biggest for the baseline model which can be explained by the memory alignment issues. The process time is slightly bigger for the models with the activation word repeat detection with 4% increase on average for the worst case of $w_{\max} = 400$.

2.3.5. Discussion of Experimental Results of the Proposed Method

In the case of false rejection, voice activation users tend to repeat the activation word. The experiments showed that a voice-activated system could leverage it to improve the false reject rate by detecting such repeats. Moreover in case of phoneme-based keyword spotters, this can be done without retraining the acoustic model and with a negligible increase in resource consumption (Table 2.18).

There is a question of whether the results can be better with the actual retraining of the model. It can be hypothesized that it can be true because the users tend to repeat the word slower and more distinctly, so the special model might detect such repeats better than the common one.

Also, it was shown that there is a tradeoff between resource consumption and the value of $w_{\max}$. The bigger the value of the $w_{\max}$, the slower the voice-activated system works (though the increase is small compared to the time of the acoustic model inference), and the better the results.

The proposed method can be used in situations when the change of the keyphrase is not possible. If there's such a possibility, making the keyphrase longer may improve the quality of detection more significantly.

In this section, the pause between keyword repeats is not modeled. It can be hypothesized that detecting and measuring the duration of such events might improve the detection quality even further.

2.4. Conclusions of Chapter 2

1. Incorporating prior knowledge into acoustic feature pipeline is justified for the current state of voice activation systems, which is shown by the

comparison of three popular acoustic feature pipelines: Mel frequency cepstral coefficients (MFCC), log Mel-filterbank energy (LFBE), and audio spectrogram. MFCC provides up to 0.4% of an accuracy boost compared to LFBE, depending on the batch size and the number of used filters, but the globally best accuracy is the same and equal to 80.7%. At the same time, the globally best accuracy for the spectrogram is 67.5%.

2. MFCC computation requires on average 16% more CPU consumption than LFBE, which introduces a tradeoff between the spotting quality and resource distribution.
3. Fine-tuning of frame length in the feature extraction pipeline of a voice activation system allows getting 0.95% ± 0.77% (95% confidence interval) improvement in the detection accuracy for the Google Speech Commands dataset (Warden, 2018) and convolutional neural networks compared to the default values of 20 ms and 30 ms.
4. Two new acoustic units were proposed for the keyword spotting task: a uniform unit and an adaptive unit.
5. The experiments showed that the use of phonemes as an acoustic unit was the best default option, especially in a low-resource setup; the proposed uniform target was the second on average (around 30% behind) but showed better results in the case of a big number of training examples.
6. The detection accuracy is 11.78% ± 7.77% (95% confidence interval) higher when all the occurrences of the acoustic unit present in the training set are used as targets, compared to the option of using only the occurrences of the acoustic unit inside the keyword.
7. The method to improve the false reject rate of a voice-activated system in a practical setup was proposed. The method allows for the detection of the repeats of an activation word (which happens in the case of a false rejection) without retraining the acoustic model and with only a negligible growth in resource consumption: the process time is increased by 4% on average.
8. In the experiments, the method allows detecting 13.32% ± 1.76% (95% confidence interval) of previously undetected repeats of the activation word while not adding a single false alarm on the hold-out dataset. It is also possible to detect 79.9% ± 1.5% (95% confidence interval) of repeats while adding less than 1% of the false alarm rate. This proves that the detection of keyword repeats improves the quality of a voice-activated system.

3

Training a Voice-Activation System in a Low-Resource Setup

Voice activation systems solve the task of a finding predefined keyword or keyphrase in an audio stream (Chapter 1). Since the 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.

One of the problems of the current state-of-the-art solutions is the use of very large datasets for training a neural network. E.g., Chen et al. (2014a) used hundreds of thousands of samples per keyword, and Jose et al. (2020) used millions. On the other hand, there is no clear way to build a high-quality voice activation system when the training data is sparse, for example, for low-resource languages, such as Lithuanian. This presents the question of how to build a high-quality voice activation system in cases when limited training data is available. This can be useful for:

- 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.

This chapter proposes and investigates several known and new ways to improve the quality of a voice activation system on a low-resource labeled dataset compared to the standard supervised way of acoustic model training. In this chapter, a dataset with no more than 20 training samples for each keyword is called a low-resource or small dataset. A small Lithuanian dataset is used to compare investigated methods.

In all experiments the same neural network architecture is used, but some changes are introduced to audio features (Section 3.2.4), use different kinds of pre-training (Section 3.2.6 and Section 3.2.7), or modify the training procedure (Section 3.2.8 and Section 3.2.9). Specifically, the wav2vec method (Schneider et al., 2019) is used. It was shown that it could improve the quality of the resulting system if there are less than 20 samples per keyword for several datasets, namely Google Speech Commands (Warden, 2018), a private Russian dataset, and a novel Lithuanian dataset, even though wav2vec model was trained only on English recordings. It was also shown that the use of unsupervised pre-trained audio features and especially joint-training with a high-resource dataset from another domain could significantly outperform a strong baseline of fine-tuning a model trained on another dataset. A relative improviement of 7–25% was achieved depending on the model architecture. Finally, the best test accuracy on the Lithuanian dataset was further improved from 90.77% to 93.85% by applying the joint-training of the acoustic model on two datasets of different languages: English and Lithuanian.

The results of these findings were published in the international journal (Kolesau and Šešok, 2020c, 2021b).

3.1. Collecting Lithuanian Speech Commands Dataset

To boost voice-activation research in Lithuanian, a dataset in the format of the Google Speech Commands dataset (Warden, 2018) was prepared. This section describes how the data collection and preparation were carried out.

Firstly, 20 keywords were chosen to be recorded by translating some of Google Speech Commands (Warden, 2018): “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). The words “ne,” “ačiū,” “stop,” “įjunk,” “išjunk,” “į viršų,” “į apačią,” “į dešinę,” “į kairę,” “startas,” “pauzė,” “labas,” and “iki” were selected as target classes for the voice activation problem. These words were chosen to ensure a challenging problem of differentiating words with similar word parts: compare “įjunk” and “išjunk;” the starts of keyphrases “į viršų,” “į apačią,” “į dešinę,” “į kairę;” the ends of “startas” and “labas.” Concise keywords “iki” and “ne” also increase the complexity of the task.

28 volunteers (15 males, 13 females, all aged 20–50) were asked to record these words in the specified order on their mobile devices. The speed of the pronunciation was not restricted, but the volunteers were asked to make pauses between words.

28 records were collected, ranging from 24 to 64 seconds. All volunteers filled out forms given in Annexes B and C. All the records were checked for the correct order of words.

The next step was to segment these records into one-second samples. The Audacity[1] software was used in the following way:

- apply the Sound Finder analysis tool with default parameters (Fig. 3.1);
- 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.

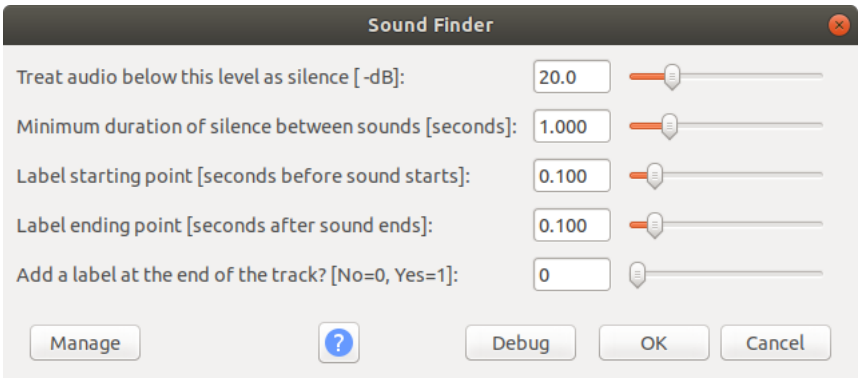


Fig. 3.1. Parameters of the Sound Finder analysis tool

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

- skip the current segment if $e_i - s_i > 1$, because the word itself is longer than 1 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 ($\varepsilon = 0.1$ to make at least a little pause before

[1]The Audacity® software©, 1999–2020, by Audacity Team. The name Audacity® is a registered trademark of Dominic Mazzoni.

the start of the word, $e_0 = 0$, s_{i+1} for the last segment is equal to the whole utterance duration):

$$A_i = \max(e_{i-1}, \min(s_{i+1} - 1 - \varepsilon, s_i - \varepsilon)), \tag{3.1}$$

$$B_i = s_i - \varepsilon; \tag{3.2}$$

- 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 longer than one second, exactly one one-second sub-segment was uniformly picked and used as background noise. 292 non-speech segments were received in such a way.

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

3.2. Empirical Evaluation of the Methods for Training a Keyword Spotter in a Low-Resource Setup

3.2.1. Datasets

The following three datasets were used in the experiments:

- English dataset – the Google Speech Commands dataset (GSC) (Warden, 2018);
- Russian dataset – a private dataset;
- Lithuanian dataset – see Section 3.1 for details.

The Google Speech Commands dataset (Warden, 2018) is the same dataset that was used in Section 2.1. For more information, see Section 2.1.3.

The Russian dataset is private and 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 the samples from the Google Speech Commands dataset (Warden, 2018) were reused. The following 12 classes were chosen: “один” (one), “два” (two), “три” (three), “четыре” (four), “пять” (five), “да” (yes), “нет” (no), “спасибо” (thanks), “стоп” (stop), “включи” (turn on), unknown and silence. Note that this dataset is not the same as in Section 2.2 because the phoneme positions for each keyword were needed in Section 2.2. Also, this dataset is not the same as in Section 2.3 because the keyword repeats were needed in Section 2.3.

Also, specifically for this and the future efforts in voice activation in Lithuanian, a similar dataset was collected for the Lithuanian language. The collecting

methodology and the dataset description are given in Section 3.1. The following 15 classes were chosen: “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.

3.2.2. Neural Network Architectures

Two types of audio features and two types of neural network architectures were used in the experiments. Either log Mel-filterbank energy (LFBE) or pre-trained audio features from wav2vec model (Schneider et al., 2019) were chosen as audio features.

The log Mel-filterbank energy features are often chosen for building voice activation or a speech recognition system (Section 1.2.1). Kaldi (Povey et al., 2011) implementation of feature computation was used with the following parameters: frame width – 25 ms, frame shift – 10 ms, and the number of bins – 80. A 98×80 feature matrix is the result of computing LFBE on a one-second sample from datasets. The method torchaudio.compliance.kaldi.fbank can be used in PyTorch (Paszke et al., 2019) to reproduce this computation.

In the case of wav2vec audio features, the pre-trained model from https://github.com/pytorch/fairseq/blob/master/examples/wav2vec/README.md was used. In that case, a 98×512 feature matrix is an input for the neural network.

The following neural network architectures were used: a three-layer fully connected neural network and residual neural networks (resnet), as described by Tang and Lin (2018).

The used fully connected neural network consists of the following blocks:

- a fully connected layer of the size 128;
- a rectified linear unit (ReLU) as an activation function (Nair and Hinton, 2010);
- a fully connected layer of the size 64;
- ReLU;
- flattening of the $T \times 64$ matrix in the $64T$ vector, where T is the number of frames in the sample (98 in all the experiments);
- a fully connected layer of the size C, where C is the number of classes to discriminate;
- a softmax-layer.

This neural network architecture is presented in Fig. 3.2.

Resnets used in the experiments are based on work by He et al. (2016) and repeat the solutions by Tang and Lin (2018). He et al. (2016) proposed that it may be easier

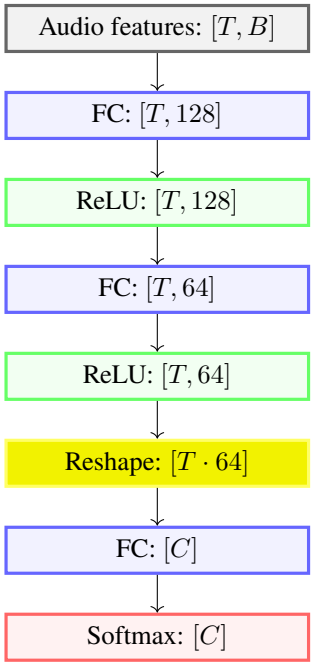


Fig. 3.2. Scheme of feedforward neural network used in the experiments. The shape of the resulting tensor is specified after each block in square brackets. FC stands for a fully-connected layer. T is the number of frames in a sample (98 in all the experiments), B is the number of channels in audio features (80 for log-mel filterbanks, 512 for wav2vec features), C is the number of classes to discriminate

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, where the input of some layer is added to the output of some downstream layer. See Fig. 3.3 for the whole network architecture and Fig. 3.4 for the detailed architecture of the residual block. 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 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 time and frequency dimensions, making them smaller, and the second such layer reduces them to 1, essentially removing these dimensions.

Architectures by Tang and Lin (2018) consist of the following blocks:

- a bias-free 3 × 3 convolutional layer;
- an optional average pooling layer (e.g., 4 × 3 layer in res8);

- several residual blocks consisting of repeated 3×3 convolutions; ReLU and batch normalization layers (Ioffe and Szegedy, 2015) (Fig. 3.4);
- a 3×3 convolutional layer;
- a batch normalization layer;
- an average pooling layer;
- a fully connected layer of size C, where C is the number of classes to discriminate;
- a softmax-layer.

All the layers are zero-padded. For some variants, dilated convolutions are applied to increase the receptive field of the model. Parameters of all used resnet architectures can be seen in Table 3.1. Dilation is the technique to introduce gaps in the convolution filters to get more spatial information with the same number of parameters.

Table 3.1. Parameters of resnet architectures used in experiments. The names follow Tang and Lin (2018)

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

3.2.3. Neural Network Training

The experiments followed the same identical procedure as the TensorFlow reference for the Google Speech Commands dataset (Warden, 2018). The Speech Commands Dataset (Warden, 2018) was split into training, validation, and test sets, with 80% for training, 10% for validation, and 10% for test. This resulted in roughly 80 000 examples for training and 10 000 each for the validation and testing. For the Russian dataset, these numbers were roughly 320 000 and 40 000. For the Lithuanian dataset, 300 records were used for training, 75 for validation, and 114 for testing (the distribution was skewed to ensure more stable test results). For consistency across runs, the SHA1-hashed name of the audio file from the dataset determines the split.

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

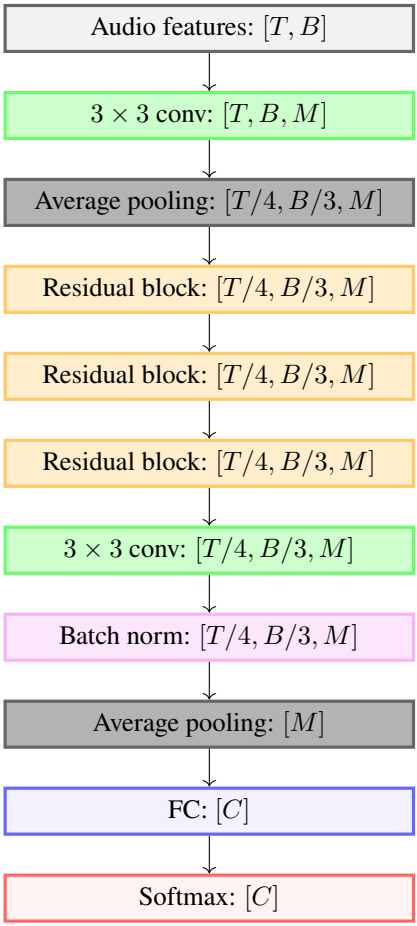


Fig. 3.3. Scheme of the res8 neural network used in the experiments. See Fig. 3.4 for the scheme of each residual block

Accuracy is the main metric for the current research. For each instance, the model outputs its most likely prediction, and is not given the option of "do not know."

An extensive random hyperparameter search (Bergstra and Bengio, 2012) was run for all experiments to reliably compare audio features and architectures. The stochastic gradient descent (SGD) was used with an initial learning rate L, momentum 0.9, mini-batch size BS (see Sections 3.2.4–3.2.9 for specific values of hyperparameters). The validation metrics (cross-entropy (CE) loss and accuracy) are computed every S steps of optimization. A minimal validation accuracy is stored. If the new validation accuracy is bigger than the minimal or if the cross-entropy

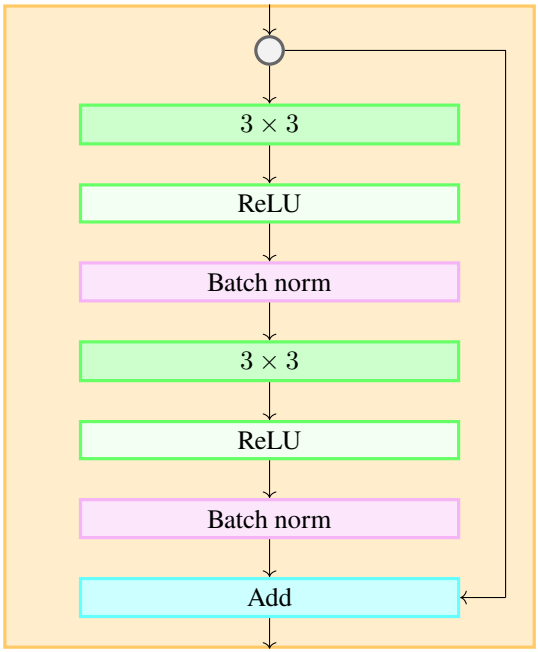


Fig. 3.4. Detailed schema of each residual block. See Fig. 3.3 for the whole neural network architecture

loss gets a "not a number" value, the weights of the best (by the validation accuracy) step are loaded, but the learning rate drops by the factor of L'. The training process stops when the learning rate drop happens for the 6^{th} time. The test accuracy is computed exactly once: on the best model, by the validation accuracy, at the end of the training process. The test accuracy is reported in this work.

The values were chosen as follows:

- $-\log_{10} L$ from $U\{0, 3\}$;
- $\log_2 \text{BS}$ from $U\{4, 7\}$;
- $\log_2 S$ from $U\{3, 12\}$;
- L' from $U\{1.1, 10.0\}$.

U is a uniform distribution (discrete uniform distribution in the case of S and BS).

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

For each experiment a random hyperparameter search was conducted (Section 3.2.3) and the best models on the validation set were chosen. Note that only the test accuracy (averaged over three runs) was reported for all the methods to not clutter the description.

First, the baseline metrics were established for all the datasets in Section 3.2.4. All the next subsections of this Section aim to improve the results on the small Lithuanian dataset.

For more discussion of the results, see Section 3.3.

3.2.4. Baseline Metrics for all Datasets

To get baseline metrics experiments on full datasets were run with both log Mel-filterbank energy and wav2vec features. The best results of these runs are presented in Table 3.2 for the English dataset.

The same experiments were repeated for the Russian dataset and received similar results (Table 3.3): ff and resnet8-narrow as the simplest models received better results with wav2vec audio features. But the overall best result was still with LFBE: 97.22%. The best result of wav2vec runs is 96.62% which is worse but still very competitive.

Notably, the wav2vec model was trained on the Librespeech dataset (Panayotov et al., 2015) containing only English audio books. It is promising that this model allows for a good accuracy both for Russian and Lithuanian datasets (Table 3.4). Moreover, results on the Lithuanian dataset using wav2vec are better than using LFBE (90.77% vs 89.23%).

Table 3.2. Test accuracy on the full English dataset

Architecture	LFBE	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%

3.2.5. Unsupervised Pre-training for a Low-Resource Setup

To get insights, experiments were run with small amount of training data coming from different datasets. The number of training samples per keyword was limited to 3, 5, 7, 10, and 20 for English and Russian datasets. The size of validation and test sets remained the same to get reliable and comparable results. These experiments are motivated as follows. First, Schneider et al. (2019) reported that they achieved state-of-the-art results in automatic speech recognition with unsupervised pre-training in the case when limited train data was available. Second, the first set of experiments

Table 3.3. Test accuracy on the full Russian dataset

Architecture	LFBE	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 3.4. Test accuracy on the Lithuanian dataset

Architecture	LFBE	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%**

showed that wav2vec audio features were superior when the machine learning model was simple. Simpler models tend to perform better when a dataset is small. So, it might be beneficial to use unsupervisedly pre-trained audio features in this scenario. The results of these experiments are summarized in Table 3.5.

Table 3.5. Test accuracy on limited datasets

Limit	Dataset	LFBE	wav2vec	Relative improvement
3	en	39.30%	37.05%	-5%
3	ru	31.72%	48.51%	53%
5	en	45.91%	59.41%	29%
5	ru	40.42%	57.65%	42%
7	en	50.55%	60.41%	20%
7	ru	57.69%	63.88%	11%
10	en	54.40%	61.05%	12%
10	ru	54.60%	58.68%	7%
20	en	74.22%	75.63%	2%
20	ru	67.43%	73.79%	9%

The choice of hyperparameters is provided in Annex A for the results of this Section to make it easier to reproduce the findings. See Table A2 for the results on the Google Speech Commands dataset (Warden, 2018), Table A3 for the results on

the Russian dataset and Table A1 for the results on the Lithuanian dataset (Annex A).

To statistically support the claim regarding the better choice of wav2vec models for the limited trainsets, confidence intervals were computed. For each choice of the dataset and limit (5 or 7), 25 experiments were run. Unfortunately, it was not possible to use higher values of limits for the Lithuanian dataset because of its small size. In each experiment, the corresponding number of train utterances was chosen (unlike stated in Section 3.2.3, where the choice was fixed for the sake of reproducibility). For each dataset and limit, the best architecture was chosen judging by previous experiments. For each experiment the accuracy gain of switching from LFBE to wav2vec was saved. The resulting test accuracies were processed with the bootstrap method from SciPy library (Virtanen et al., 2020) to compute the 95% confidence interval. The results are presented in Fig. 3.5.

As demonstrated, each language has a statistically confident gain for the limit of five utterances (confidence intervals are not overlapping) and no statistically confident gain for the limit of seven utterances. All the gains were also processed with the bootstrap method. The results are summarized in Table 3.6. As demonstrated, confidence intervals are more modest than the values from Table 3.5.

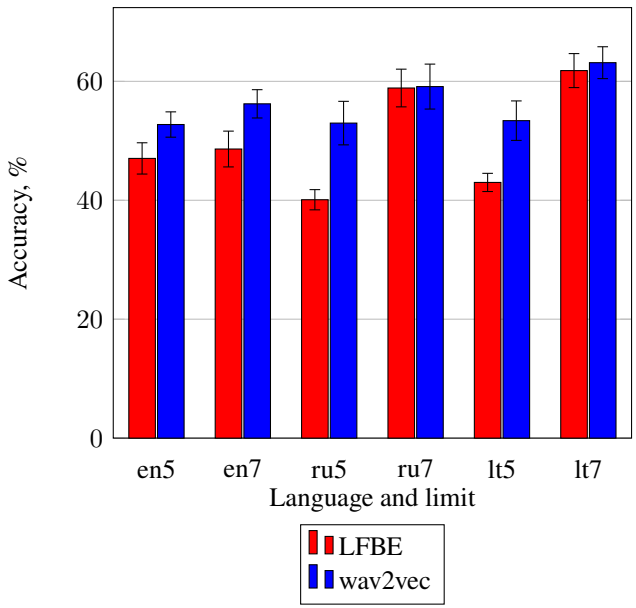


Fig. 3.5. Test accuracy for LFBE and wav2vec models for a limited trainset and each dataset with confidence intervals

Table 3.6. Confidence intervals for the accuracy gain for limited datasets and switching from LFBE to wav2vec

Limit	Mean accuracy gain	Confidence interval
5	24.68%	(18.44%, 30.93%)
7	8.04%	(3.31%, 12.76%)

3.2.6. Fine-Tuning the Pre-Trained Model

Starting from this subsection, the goal of the research is to improve the result on the small Lithuanian dataset. The results from Table 3.4 are used as the baseline metrics.

Fine-tuning the model which was pre-trained on a large dataset is a popular way to improve model performance in a specific domain or task (Ge and Yu, 2017; Li et al., 2020a). This method is widely used in computer vision problems (Chen et al., 2018; Ren et al., 2015) but also can be used in voice-related problems, e.g., speech recognition (Bansal et al., 2019).

The general framework is as follows:

- train the classifier on a bigger dataset with the random weights initialization;
- uptrain the classifier on the target dataset.

The general idea is that the classifier can learn implicit features on a bigger dataset that can be useful for a target dataset, even though the pre-training dataset may be out-of-domain. Sometimes, some parameters must be changed before the uptraining, e.g., if the target and original datasets have a different number of classes. In that case, at least the final fully-connected layer in the models must be reinitialized.

The Google Speech Commands dataset (Warden, 2018) is used for the pre-training. The log Mel-filterbank energy (LFBE) features were chosen as input features. The hyperparameter grid search was conducted for three model architectures, and the best-performing models were chosen for the fine-tuning. The training procedure is explained in Section 3.2.3. The test accuracy on the Google Speech Commands dataset (Warden, 2018) is presented in Table 3.7 for the selected models.

Table 3.7. Test accuracy for models pre-trained on the Google Speech Commands dataset (Warden, 2018)

Architecture	Accuracy
res8	95.07%
res15	97.03%
res26-narrow	97.46%

Then, neural networks were trained on the Lithuanian dataset following the procedure from Section 3.2.3, but the weights were initialized using the pre-trained

models except for the final linear layer, which was initialized randomly. A hyper-parameter grid search was also performed, and the resulting metrics for the best models are presented in Table 3.8.

Table 3.8. Test accuracy after the fine-tuning on the target dataset. The best result for each architecture is highlighted

Architecture	Baseline accuracy	Method accuracy
res8	78.46%	**89.23%**
res15	89.23%	**90.77%**
res26-narrow	83.08%	**89.23%**

3.2.7. Exemplar-Like Pre-Training

The kind of pre-training discussed in this subsection is based on the pre-training suggested by Dosovitskiy et al. (2014) for image classification. It was proposed to adapt this method for a voice activation problem. The authors of the (Dosovitskiy et al., 2014) proposed to learn (unsupervisedly) features that were invariant to some predefined transformations as follows:

- select n "interesting" (seed) samples;
- generate $k-1$ samples from each of the samples using the augmentations (so the dataset of nk samples is built): the source sample and $k-1$ augmented samples form a surrogate class;
- train a classifier to distinguish n surrogate classes.

The resulting classifier can be fine-tuned to the downstream task. "Interesting" samples mean that the chosen sample should contain information similar to the samples of the downstream task. For example, Dosovitskiy et al. (2014) choose patches of images with a considerable gradient, so these patches contain objects or part of objects.

Carefully chosen augmentations should ensure that the resulting classifier would be invariant for some transformations. For example, Dosovitskiy et al. (2014) used scaling, translation and rotation, so the classifier did not depend on the position of the classified object, just on the object itself.

Two kinds of augmentations were used: the common for the Google Speech Commands dataset (Warden, 2018) described in Subsection 3.2.3 and SpecAugment (Park et al., 2019). The latter consists of combination of the following transformations (see (Park et al., 2019) for a detailed explanation):

- time warping;
- frequency masking;
- time masking.

The visualization of all these transformations is presented in Fig. 3.6.

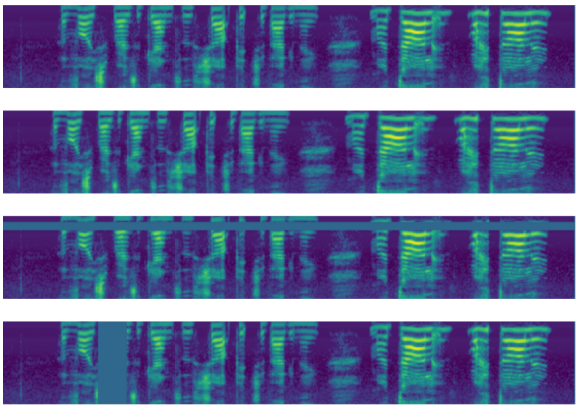


Fig. 3.6. LFBE visualization after applying time warp (the 2nd image), frequency masking (3rd image) and time masking (the 4th image)

The severity of transformations is controlled via the set of hyperparameters. The values used in the experiments are presented in Table 3.9.

Table 3.9. Hyperparameter values for used SpecAugment levels

Parameter / Level	1	2	3
Maximal number of frames in frequency mask	25	15	40
Maximal number of frequencies in time mask	25	15	40
Number of frequency masks	1	2	1
Number of time masks	1	2	1

The level of SpecAugment transformations was a hyperparameter in all grid searches in this subsection.

Two sets of experiments were conducted using this kind of pre-training. First, the best baseline classifier (res8 trained on wav2vec features) was run on 20 hours of lectures in Lithuanian. More specifically, the classifier was applied to one-second patches of each audio file with a 100 ms shift between the patches. Then, top K patches were selected by the model score for each of the classes. It was made sure that no two patches were closer than 10 seconds apart (so there were no too similar samples). Those patches were the seed samples for the first set of experiments. Several values of K were tried, so the number of seed samples was 100, 1000, and 2000. Then, a grid search was conducted for several architectures. The resulting metrics of the pre-training are presented in Table 3.10 (only models with the testing accuracy equal to or higher than 90% were left).

Table 3.10. Test accuracy of the pre-trained classifiers on surrogate samples

Architecture	Seed samples	SpecAugment level	Accuracy
res8	100	1	96.00%
res8	100	2	100.00%
res8	100	3	99.00%
res8	1000	3	97.00%
res15	2000	1	92.40%
res15	2000	3	90.00%
res8	2000	2	97.10%

The weights of these models were used for initialization (except for the last linear layer) for fine-tuning on the target Lithuanian dataset. The choice of the model was one of the hyperparameters in the grid search. The results are presented in Table 3.11.

Table 3.11. Resulting test accuracy after pre-training on surrogate samples. The best result for each architecture is highlighted

Architecture	Baseline accuracy	Method accuracy
res8	78.46%	**80.00%**
res15	**89.23%**	81.54%

Table 3.12. Test accuracy of the classifiers pre-trained on the trainset of the Lithuanian dataset

Architecture	SpecAugment level	Accuracy
res15	1	38.92%
res15	2	79.77%
res15	3	67.48%
res26-narrow	1	72.22%
res26-narrow	2	5.76%
res26-narrow	3	22.79%
res8	1	1.28%
res8	2	83.48%
res8	3	82.33%

The second set of experiments is the same as the first, except for the choice of the seed samples. 200 audio files from the trainset of the target dataset were chosen as source samples. It was hypothesized that pre-training on the samples from the target domain would improve the quality of the resulting classifier. The results of pre-training are presented in Table 3.12. The pre-training metrics are somewhat poor; it was even impossible to get a 10% accuracy for some setups.

Still, even 1.28% that was received for res8 and the 1st level is better than a random guess (0.5%). It can be hypothesized that the results are worse than in Table 3.10, because seed samples in this set of experiments are much closer to one another, some because of the same speaker, and some because of the same spoken word.

As in the first set of experiments, the resulting models were fine-tuned on the target dataset. The results are presented in Table 3.13.

Table 3.13. Resulting test accuracy after pre-training on the trainset of the Lithuanian dataset. The best result for each architecture is highlighted

Architecture	Baseline accuracy	Method accuracy
res8	**78.46%**	75.38%
res15	89.23%	**90.77%**
res26-narrow	**83.08%**	75.38%

3.2.8. Self-Training

Self-training is the method of semi-supervised training when the baseline (teacher) model is used to label available unlabeled data (Triguero et al., 2015). These labels are called pseudolabels. The samples with pseudolables are appended to the supervised data, and the target (student) model is trained on the resulting dataset. The expected profit comes from the fact that the student model is learning to get the same results on the augmented (corrupted) samples as the teacher model on the original (not corrupted) samples. Self-training shows good results for speech recognition (Kahn et al., 2020).

The same set of lectures was used as in Subsection 3.2.7 for getting the unlabeled data. The baseline models were used to get the top 1000 samples for each class by the model score, making sure that all samples were not closer than ten seconds to one another. All the samples with pseudolabels are added only to the training set. The validation set and the test set are remained the same to make a fair comparison to the baseline models. The choice of the baseline model for the pseudolabel generation is a hyperparameter in a grid search in this experiment. The results are presented in Table 3.14.

3.2.9. Joint Training on Two Datasets of Different Languages

A good classifier should be invariant to some transformations. It might be difficult to learn these patterns in low-resource settings. Using a bigger annotated dataset could help with this problem if it is out of domain. In this case (and if the collecting of such a dataset is not difficult), it can be used to improve the resulting quality on a small target dataset.

Table 3.14. Test accuracy after the training with the initial dataset and added samples with pseudolabels. The best result for each architecture is highlighted

Architecture	Baseline accuracy	Method accuracy
ff	**72.31%**	63.08%
res8-narrow	73.85%	**75.38%**
res8	78.46%	**81.54%**
res15-narrow	83.08%	**84.62%**
res15	**89.23%**	86.15%
res26-narrow	**83.08%**	58.46%
res26	80.00%	**81.54%**

A set of experiments was conducted on using the Google Speech Commands dataset (Warden, 2018) to make a better classifier for the Lithuanian dataset. First, all the samples from the bigger dataset were added to the Lithuanian as unknown samples and trained the classifier on the resulting dataset. The results are presented in Table 3.15.

Table 3.15. Test accuracy after using the Google Speech Commands dataset (Warden, 2018) as additional unknown samples. The best result for each architecture is highlighted

Architecture	Baseline accuracy	Method accuracy
ff	**72.31%**	66.15%
res8-narrow	**73.85%**	67.69%
res8	**78.46%**	76.92%
res15-narrow	**83.08%**	80.00%
res15	**89.23%**	80.00%
res26-narrow	**83.08%**	55.38%
res26	**80.00%**	76.92%

Next, the classifiers were trained with the task to distinguish both target words from the Google Speech Commands dataset (Warden, 2018) and the Lithuanian dataset. The results are in Table 3.16 and Table 3.17.

Finally, the previous experiment was repeated, but every Lithuanian sample in the training set was repeated T times for the samples of different languages to make a comparable influence on the training process. $T = 10$ showed the best results presented in Table 3.18.

To statistically support the claim of a better choice of joint training, confidence intervals were computed. For each choice of architecture, 25 experiments were run. For each experiment, the accuracy gain of switching from the baseline method to the joint training was saved. The resulting test accuracies were processed with the bootstrap method from the SciPy library (Virtanen et al., 2020) to compute a 95% confidence interval. The results are presented in Fig. 3.7.

Table 3.16. Test accuracy on both Lithuanian and English commands after using the Google Speech Commands dataset (Warden, 2018) as additional target samples

Architecture	Accuracy
ff	93.01%
res15	97.85%
res15-narrow	97.67%
res26	97.31%
res26-narrow	94.98%
res8	96.77%
res8-narrow	96.06%

Table 3.17. Test accuracy on Lithuanian commands after using the Google Speech Commands dataset (Warden, 2018) as additional target samples. The best result for each architecture is highlighted

Architecture	Baseline accuracy	Method accuracy
ff	**72.31%**	33.33%
res8-narrow	**73.85%**	19.44%
res8	**78.46%**	47.22%
res15-narrow	**83.08%**	20.83%
res15	**89.23%**	62.50%
res26-narrow	**83.08%**	19.44%
res26	**80.00%**	54.17%

All the gains were also processed with the bootstrap method. The resulting average gain is 10.17% ± 2.11% (95% confidence interval).

To check whether the provided solutions could be used for voice activation, the CPU consumption was measured. The time needed to process a 100-second audio file was measured for any given model and acoustic feature pipeline 100 times. One core of Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz was used for testing. The average time consumption is provided in Table 3.19.

3.3. Discussion of Experimental Results of the Investigated Methods

The baseline results for the full English dataset are presented in Table 3.2. Note that slightly better results were received compared to those by Tang and Lin (2018) in the LFBE setting. It can be explained by the following reasons:

- the Google Speech Commands dataset (Warden, 2018) was extended since their publication;
- a more extensive hyperparameters search was performed.

Table 3.18. Test accuracy after using the Google Speech Commands dataset (Warden, 2018) as additional target samples and repeating each Lithuanian sample ten times. The best result for each architecture is highlighted

Architecture	Baseline accuracy	Method accuracy
ff	72.31%	**84.62%**
res8-narrow	73.85%	**89.23%**
res8	78.46%	**84.62%**
res15-narrow	83.08%	**89.23%**
res15	89.23%	**93.85%**
res26-narrow	83.08%	**90.77%**
res26	80.00%	**90.77%**

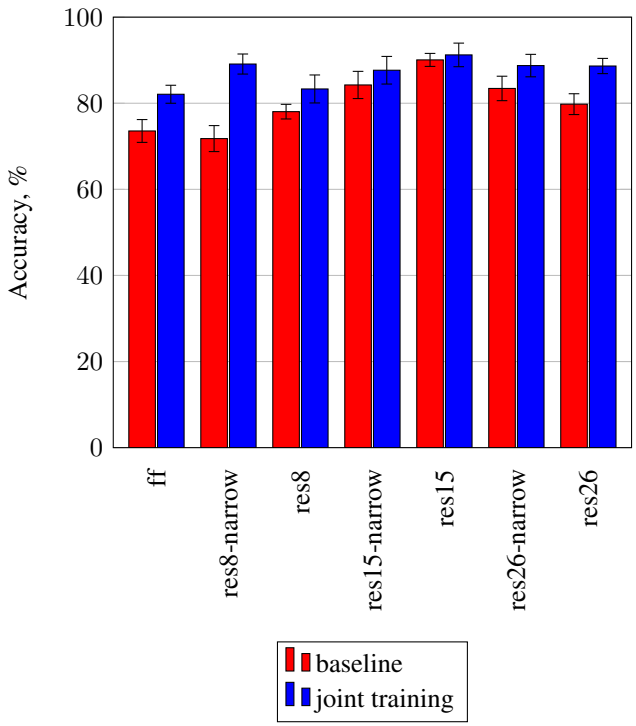


Fig. 3.7. Confidence intervals for test accuracy for all tested models, compared to the baseline and joint training method

Table 3.19. Average time spent for processing a 100-second audio

Architecture / Acoustic Features	LFBE	wav2vec
ff	0.13s	12.31s
res8-narrow	0.30s	12.44s
res8	0.89s	12.96s
res15-narrow	1.12s	13.45s
res15	5.97s	15.71s
res26-narrow	1.01s	13.56s
res26	3.12s	14.36s

The following conclusions can be made from the results:

- wav2vec audio features give a competitive result for the voice activation problem with very simple downstream models such as feedforward neural network;
- the benefit of unsupervised pre-training vanishes as the model gets more sophisticated and deep.

Table 3.3 shows that the results on the Russian dataset are similar to the results of the Google Speech Commands dataset (GSC) (Table 3.2). Specifically, in the case of the smallest models (ff, res8-narrow), wav2vec acoustic feature pipeline provides better results than LFBE. In the case of bigger models, results are close, but LFBE is generally preferable. It is interesting that for the Russian dataset, the global best results are not for the biggest model (res-26) but for res-15 which suggests overfitting. Overall, similar values of accuracy for the Russian and English datasets suggest that the results are reproducible for other European languages. Note that the results for Lithuanian (Table 3.4) are worse, because the training dataset is much smaller (so the results are understandably worse).

As demonstrated, the use of pre-trained audio features, such as wav2vec, increases the system accuracy by approximately 10% when up to ten samples are used per keyword both for English and Russian languages (Table 3.5) even though that the model was only trained on English audio records. The increase is even bigger if five samples are used. And it almost vanishes if up to 20 samples are used. Great results on small extracts of English and Russian datasets and on the full Lithuanian dataset (Table 3.4) show that the wav2vec features (and the unsupervised pre-training in general) are well suited for the voice activation problem in the case of small datasets.

Table A2, Table A3, and Table A1 (Annex A) suggest the following points:

- the value of cross-entropy loss generally agrees with the value of the test accuracy, which suggests that at least for the used datasets, the optimizing loss is adequate for maximizing accuracy;

Table 3.20. Test accuracy from the experiments of this work. The results that are better than log-mel filterbanks baseline are highlighted

Method / Architecture (Section)	ff	res8-narrow	res8	res15-narrow	res15	res26-narrow	res26
LFBE (3.2.4)	72.31%	73.85%	78.46%	83.08%	89.23%	83.08%	80.00%
wav2vec (3.2.4)	**78.46%**	**84.62%**	**90.77%**	80.00%	86.15%	72.31%	**83.08%**
Fine-tuning (3.2.6)	N/A	N/A	**89.23%**	N/A	**90.77%**	**89.23%**	N/A
Exemplar on surrogate (3.2.7)	N/A	N/A	**80.00%**	N/A	81.54%	N/A	N/A
Exemplar on trainset (3.2.7)	N/A	N/A	75.38%	N/A	**90.77%**	75.38%	N/A
Self-training (3.2.8)	63.08%	**75.38%**	**81.54%**	**84.62%**	86.15%	58.46%	**81.54%**
GSC as unknown (3.2.9)	66.15%	67.69%	76.92%	80.00%	80.00%	55.38%	76.92%
Added GSC (3.2.9)	33.33%	19.44%	47.22%	20.83%	62.50%	18.44%	54.17%
Added GSC with scale (3.2.9)	**84.62%**	**89.23%**	**84.62%**	**89.23%**	**93.85%**	**90.77%**	**90.77%**

- the optimal value of S tends to be bigger for full datasets than for small datasets: this is because a smaller interval between validations allows early stopping, which is crucial for small datasets to avoid overfitting;
- the interval for L and L' is adequate, it does not seem to be a way to effectively cut the search space;
- the optimal batch size in some cases hits the biggest value in the search space (64), but unfortunately, it is hard to extend the search space because bigger values of the batch size because out-of-memory errors in the GPU-memory.

Next, focus is shifted to improving the results in the case of small datasets. The results of all the experiments for the Lithuanian dataset are summarized in Table 3.20. For each architecture and for each investigated method, the test accuracy of the best model is shown (models are chosen by their performance on the validation set). As demonstrated, there are several ways to improve the baseline quality of the voice activation in a low-resource setup.

The goal of the research is to find methods of improving the detection quality of a voice activation system in a low-resource data set setup compared to standard training on the data set in a supervised manner (models with LFBE features from Section 3.2.4 and, specifically, Table 3.4).

Using unsupervised pre-trained audio features is one such way, as was discussed in Section 3.2.4 (models with wav2vec features).

Fine-tuning a model pre-trained on a similar data set but in another language improves the results as expected. Interestingly,the pre-trained models show rather different results on the English dataset (Table 3.7). The results after fine-tuning are rather close (Table 3.8). Moreover, not the best pre-trained model (in terms of accuracy) gives the best fine-tuned accuracy. It can be explained by the fact that all pre-trained models learned features that are useful for voice activation in Lithuanian, but the res26 model is a little more overfitted to English.

The exemplar-like pre-training (Dosovitskiy et al., 2014) on surrogate audio patches produces results comparable to the baseline but is unable to get a significant margin. The pre-training accuracies on Lithuanian set of lectures are presented in Table 3.10. As demonstrated, the pre-training task is rather simple, especially for smaller models (res8), which are more robust to overfitting. On the one hand, high accuracy values are good, but on the other hand, they might mean that the models do not actually learn good enough features. It is confirmed by fine-tune results in Table 3.11, where for some models, even the degradation of the quality can be observed. It can be hypothesized that the limited improvement is explained by the fact that the chosen augmentations do not cause invariance to the change of a speaker, which is crucial for voice activation. There was also an attempt to pre-train the model on the target training set (Table 3.12) and obtain some improvement for some models but a quality loss for others (Table 3.13). On the one hand, the pre-training dataset is much closer to the target domain. On the other hand, it is not obvious that pre-training task is useful in that setting: all the features needed to distinguish the same words from different speakers (or even different pronunciations of the same speaker) are not immediately useful for keyword spotting.

Self-training also helps, especially for the models with a lower capacity (Table 3.14). Still, using wav2vec is almost always better, possibly because of not very useful datasets for pseudolabels generation: there is no actual pronunciation for more than half of the target words in the lecture data set. It is not just an experiment drawback but also a practical limitation: it is difficult to find a good source dataset for pseudo-labels if the target words are rare. In other cases, the investigation of iterative pseudolabeling (Xu et al., 2020) might be an interesting choice for future research.

Finally, an attempt was made to use the Google Speech Commands dataset (Warden, 2018) in the training process, not only as a pre-training dataset. It was hypothesized that adding the dataset as unknown samples would help the classifier

to learn more patterns, but there was no success (Table 3.15). There was no surprise because such addition only teaches the model to better distinguish known and unknown words but does not make it easier to distinguish between known words. Also, because of the domain shift, it is easy to learn that the audio from the English dataset always come from "unknown" class. Making English commands an additional target helps the resulting classifier (Table 3.16), but the results on only Lithuanian commands are horrible (Table 3.17) because the Lithuanian part of the data set is small compared to the Google Speech Commands part, so it is favorable for the classifier to focus on the English part. Then, the Lithuanian part was scaled up by simply repeating the samples several times. The resulting metrics are better than the LFBE baseline (Section 3.2.4) in all models and better than the wav2vec baseline (Section 3.2.4) in all models except res8. The improvement is not because of the combination of different languages but because of the use of more labeled data, which helps to learn features that are useful to distinguish different words. These features might be useful even if words are not connected or similar to target keywords. If more examples of target keywords are available, adding them to the training set should be more useful than using the dataset from another domain.

The accuracy is not the best metric for assessing the detection quality of a voice activation system because it does not reflect the balance between the two different types of errors (see Section 1.2.5 for a detailed discussion):

- False alarms;
- False rejects.

Many researchers and practitioners use the false alarm rate (FAR, the rate of false detections over the samples without a keyword) and the false reject rate (FRR, the rate of skipping the detection over the samples with a keyword). Still, the researchers that use the Google Speech Commands dataset (Warden, 2018) (and collected applying the same principles to the Lithuanian dataset) use the test accuracy as a final metric (Li et al., 2020b; Mo et al., 2020; Tang and Lin, 2018) because it is easier to compare in the case of many keywords, and because a relatively big and diverse speech dataset is required to get a reasonable estimate for the false alarm rate.

Still, it is possible to compute the FAR and the FRR even for multiclass models. The obvious method is to compute these metrics for each keyword. However, it is not very practical for the experiments: each keyword has a handful of test samples. Additionally, comparing two models by 15 pairs of false alarm rates and FRRs is not a trivial task. Consequently, the same method as the proposed in scikit-learn tutorials (Pedregosa et al., 2011) was used: this multiclass problem is converted to binary classification. For a sample with a target label t out of n classes with predicted probabilities $p_1, \ldots, p_n$, $n - 1$ negative samples are considered with predicted probabilities $p_1, \ldots, p_{t-1}, p_t, \ldots, p_n$ and one positive sample with

a predicted probability p_t. Then, the FAR and the FRR can be computed. The values of these metrics for different values of classifying thresholds are presented in Fig. 3.8 for the best models for baseline methods and joint training (Section 3.2.9).

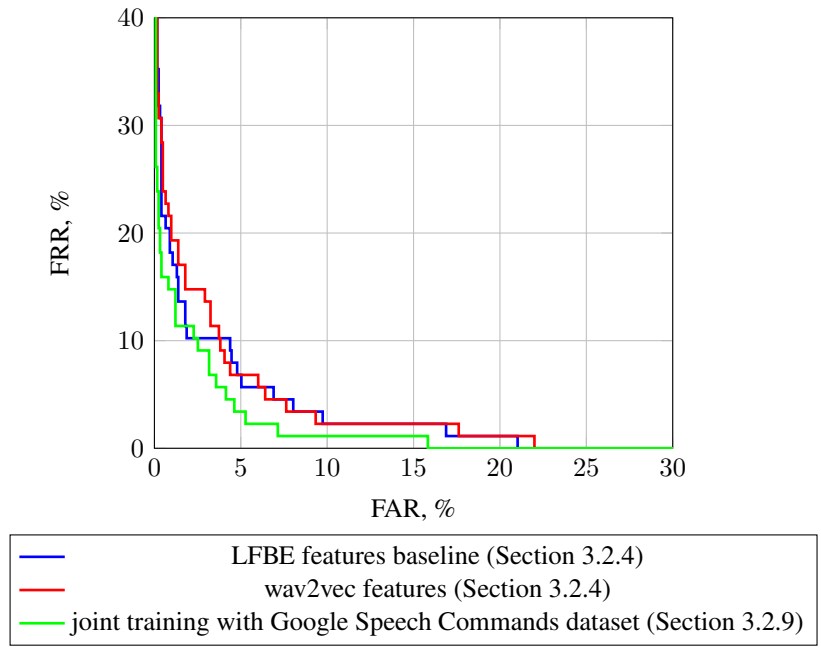


Fig. 3.8. Quality of repeat detection on operation points with FAR less than 1%

As demonstrated, the model with the highest quality (trained with joint training on both datasets) is almost everywhere better than the competitors, which supports the findings. Interestingly, the wav2vec model is better than LFBE one in terms of accuracy, it is significantly worse in operating points, with a low FAR. The equal error rate (EER, the minimal value that both FAR and FRR can accept at the same time for some value of threshold) also shows that the best LFBE model (EER $=$ 5.68%) can be better than a wav2vec (EER $=$ 5.84%) for some scenarios. The best model from the proposed joint training method has an equal error rate of 4.5%.

The confusion matrix was computed to get insights into the dataset properties and the voice activation systems behavior. 25 models were trained with the baseline method, 25 with wav2vec features and 25 with the joint training. Outputs of all these models are summarized in Fig. 3.9. The probabilities are normalized for each predicted keyword. Most of errors are concentrated on mixing known and unknown keywords. Otherwise, the most popular mistake is to predict “stop” instead of “startas,” which can be explained with the same phoneme prefix of the two words.

"Stop" and "ačiū" are often mistaken with "iki," probably because of the similar duration. Mainly, all keywords are well defined and distinguished fairly well by the models.

Actual \ Predicted	<unk>	ne	ačiū	stop	įjunk	išjunk	į viršų	į apačią	į dešinę	į kairę	startas	pauzė	labas	iki
<unk>	76	3	5	7	19	6	14	2	4	14	0	14	4	12
ne	9	93	1	1	5	0	7	4	8	4	0	4	2	0
ačiū	1	0	78	1	0	6	1	8	3	0	2	9	7	0
stop	1	0	0	85	0	13	0	0	0	2	18	0	5	4
įjunk	0	0	0	0	72	3	0	0	0	0	0	1	0	0
išjunk	0	0	0	0	0	66	2	1	1	1	0	0	0	0
į viršų	1	0	0	0	0	0	69	2	2	0	0	0	0	0
į apačią	1	0	0	0	0	0	1	67	0	2	1	0	0	0
į dešinę	0	0	0	0	0	0	3	7	68	0	1	0	0	0
į kairę	1	0	0	0	0	0	1	2	1	71	0	0	0	0
startas	0	0	0	0	0	0	0	3	1	0	75	0	1	0
pauzė	2	1	10	0	3	3	1	1	0	1	0	70	7	1
labas	3	2	0	0	0	0	1	2	10	0	1	2	74	0
iki	4	0	4	6	0	2	1	1	0	4	2	0	0	81

Fig. 3.9. Confusion matrix for the Lithuanian dataset

As demonstrated in Table 3.19, LFBE features are much faster to compute than wav2vec features. Still, the latter can be used for voice activation even on mobile devices because the whole system works 4–7 times faster than in real-time, depending on the neural network architecture. Note that although res15 and res15-narrow models are smaller than res26 and res26-narrow (Table 3.1), they are slower to use because of the absence of average pooling.

3.4. Conclusions of Chapter 3

1. In this Chapter, pre-trained audio features were proposed for voice activation systems in the case of limited training data. The experiments on the Google Speech Commands dataset (Warden, 2018) showed that these audio features improve the accuracy of the voice activation system by 8.04% ± 4.73% (95% confidence interval) when the number of samples per keyword is seven or less and by 24.68% ± 6.24% (95% confidence interval) if the number of

samples per keyword is five or less for English, Russian and Lithuanian datasets. It is also worth noting that the wav2vec model pre-trained only on English audio records was used. The improvement, however, vanishes when whole datasets are used, which may indicate limits of the proposed method.

2. Ways to improve the voice activation system quality in a low-resource language setup were adapted from other domains. Exemplar-like pre-training is the method that comes from image processing and shows results that are comparable to baseline (90.77% accuracy for res15-architecture compared to 89.23% with the baseline method, but 75.38% for res8 compared to 78.46% with the baseline method). Self-training is the method widely used in many domains. In this chapter, the approach used in automatic speech recognition was adopted for the keyword spotting task.
3. The Lithuanian dataset for future experiments on voice activation systems in a low-resource setup was collected. The experiments on this dataset showed that the use of joint training with a bigger annotated dataset produces better results than the baseline method and even fine-tuning the model trained on the high-resource dataset, but careful dataset balancing is needed. During the experiments, the proposed joint training on the Lithuanian and Google Speech Commands dataset (Warden, 2018) showed a relative 10.17% ± 2.11% (95% confidence interval) improvement for the Lithuanian part of the test-set compared to the fine-tuning. Also, the best test accuracy on the Lithuanian dataset across all architectures was achieved and is equal to 93.85%.

General Conclusions

1. The performed literature review demonstrated that designing a high-quality voice activation system is a relevant topic both for research and industry. The structure of most modern voice activation systems can be described by the following template: acoustic feature extraction pipeline (often, Mel frequency cepstral coefficients (MFCC) or log Mel-filterbank energy (LFBE) pipelines), acoustic model (almost exclusively neural networks), and the decoding stage. Most current research is focused on improving the neural network as an acoustic model part of the voice activation system.
2. Incorporating prior knowledge in acoustic feature pipeline is justified for the current state of voice activation systems, which is shown by the comparison of three popular acoustic feature pipelines: Mel frequency cepstral coefficients (MFCC), log Mel-filterbank energy (LFBE), and an audio spectrogram. MFCC provides up to a 0.4% accuracy boost compared to LFBE, depending on the batch size and the number of used filters, but the globally best accuracy is the same and equal to 80.7%. At the same time, the global best accuracy for the spectrogram is 67.5%. Fine-tuning the frame length in the feature extraction pipeline of a voice activation system allows getting a 0.95% ± 0.77% (95% confidence interval) improvement in the detection accuracy for Google Speech Commands dataset (Warden, 2018) and convolutional neural networks compared to the default values of 20 ms and 30 ms.

3. The experiments showed that the use of phonemes as an acoustic unit is the best default option, especially in a low-resource setup; the proposed uniform target is the second on average (by about 30% behind) but shows better results in the case of a big number of training examples, which can be explained by an inductive bias. Also, it was shown, that the detection accuracy was 11.78% ± 7.77% (95% confidence interval) higher when all the occurrences of the acoustic unit presented in the training set were used as targets, compared to the option of using only the occurrences of the acoustic unit inside the keyword.
4. The method for improving the false reject rate (FRR) of a phoneme-based voice activation system was proposed. It allows detecting 13.32% ± 1.76% (95% confidence interval) of previously undetected repeats of the activation word while not adding a single false alarm on the hold-out dataset. 79.9% ± 1.5% (95% confidence interval) of the repeats can also be detected while adding less than 1% to the false alarm rate. This proves that the detection of keyword repeats improves the quality of a voice-activated system. One advantage of the method is that it does not require the re-training of the neural network. The process time is increased by 4% in the worst case.
5. Pre-trained audio features were proposed for voice activation systems in the case of limited training data. The experiments on the Google Speech Commands dataset (Warden, 2018) showed that these audio features improved the accuracy of the voice activation system by 8.04% ± 4.73% (95% confidence interval) when the number of samples per keyword was seven or less and by 24.68% ± 6.24% (95% confidence interval) if the number of samples per keyword was five or less for English, Russian and Lithuanian datasets. The improvement, however, vanished when whole datasets were used, which may indicate the limits of the proposed method. Ways to improve the voice activation system quality in a low-resource setup were adapted from other domains. Exemplar-like pre-training is the method that comes from image processing and shows results that are comparable to the baseline (a 90.77% accuracy for the res15-architecture compared to 89.23% using the baseline method, but 75.38% for res8 compared to 78.46% using the baseline method). The Lithuanian dataset for future experiments on voice activation systems in a low-resource setup was collected. In the experiments on this dataset the proposed joint training on the Lithuanian and Google Speech Commands dataset (Warden, 2018) showed a relative 10.17% ± 2.11% (95% confidence interval) improvement for the Lithuanian part of the test-set compared to the baseline methods.

References

Alvarez, R., & Park, H. (2019). End-to-end streaming keyword spotting. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019* (pp. 6336–6340). IEEE.

Bahi, H., & Benati, N. (2009). A new keyword spotting approach. In *2009 International Conference on Multimedia Computing and Systems* (pp. 77–80). IEEE.

Baljekar, P., Lehman, J. F., & Singh, R. (2014). Online word-spotting in continuous speech with recurrent neural networks. In *2014 IEEE Spoken Language Technology Workshop, SLT 2014, South Lake Tahoe, NV, USA, December 7-10, 2014* (pp. 536–541). IEEE.

Bansal, S., Kamper, H., Livescu, K., Lopez, A., & Goldwater, S. (2019). Pre-training on high-resource speech recognition improves low-resource speech-to-text translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1 (Long and Short Papers)* (pp. 58–68). Minneapolis, Minnesota, Association for Computational Linguistics.

Bartkova, K., & Jouvet, D. (2015). Impact of frame rate on automatic speech-text alignment for corpus-based phonetic studies. In M. Wolters, J. Livingstone, B. Beattie, R. Smith, M. MacMahon, J. Stuart-Smith, & J. M. Scobbie (editors), *18th International Congress of Phonetic Sciences, ICPhS 2015, Glasgow, UK, August 10-14, 2015*. University of Glasgow.

Benisty, H., Katz, I., Crammer, K., & Malah, D. (2018). Discriminative keyword spotting for limited-data applications. *Speech Communication*, 99: 1–11.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13: 281–305.

Bluche, T., & Gisselbrecht, T. (2020). Predicting detection filters for small footprint open-vocabulary keyword spotting. In H. Meng, B. Xu, & T. F. Zheng (editors), *Interspeech 2020,*

21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020 (pp. 2552–2556). ISCA.

Bohac, M. (2012). Performance comparison of several techniques to detect keywords in audio streams and audio scene. In *Proceedings ELMAR-2012* (pp. 215–218). IEEE.

Chang, E. I., & Lippmann, R. P. (1994). Figure of merit training for detection and spotting. In J. D. Cowan, G. Tesauro, & J. Alspector (editors), *Advances in Neural Information Processing Systems 6* (pp. 1019–1026). Morgan-Kaufmann.

Chen, G., Parada, C., & Heigold, G. (2014a). Small-footprint keyword spotting using deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014* (pp. 4087–4091). IEEE.

Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(4): 834–848.

Chen, M., Zhang, S., Lei, M., Liu, Y., Yao, H., & Gao, J. (2018). Compact feedforward sequential memory networks for small-footprint keyword spotting. In B. Yegnanarayana (editor), *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018* (pp. 2663–2667). ISCA.

Chen, N. F., Sivadas, S., Lim, B. P., Ngo, H. G., Xu, H., Pham, V. T., Ma, B., & Li, H. (2014b). Strategies for vietnamese keyword search. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014* (pp. 4121–4125). IEEE.

Coucke, A., Chlieh, M., Gisselbrecht, T., Leroy, D., Poumeyrol, M., & Lavril, T. (2019). Efficient keyword spotting using dilated convolutions and gating. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019* (pp. 6351–6355). IEEE.

Cuayáhuitl, H., & Serridge, B. (2002). Out-of-vocabulary word modeling and rejection for spanish keyword spotting systems. In C. A. C. Coello, A. de Albornoz, L. E. Sucar, & O. C. Battistutti (editors), *MICAI 2002: Advances in Artificial Intelligence, Second Mexican International Conference on Artificial Intelligence, Merida, Yucatan, Mexico, April 22-26, 2002, Proceedings*, Vol. 2313 of *Lecture Notes in Computer Science* (pp. 156–165). Springer.

Do, C. (2019). End-to-end speech recognition with high-frame-rate features extraction. *CoRR*, abs/1907.01957.

Dogru, N., Busatlic, B., Lera, I., & Sukic, E. (2017). Smart homes with voice activated systems for disabled people. *TEM Journal*, 6: 103–107.

Dosovitskiy, A., Springenberg, J., Riedmiller, M., & Brox, T. (2014). Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1.

Dovydaitis, L., & Rudzionis, V. (2017). Identifying lithuanian native speakers using voice recognition. In W. Abramowicz (editor), *Business Information Systems Workshops - BIS 2017 International Workshops, Poznań, Poland, June 28-30, 2017, Revised Papers*, Vol. 303 of *Lecture Notes in Business Information Processing* (pp. 79–84). Springer.

Edu, J. S., Such, J. M., & Suarez-Tangil, G. (2020). Smart home personal assistants: A security and privacy review. *ACM Comput. Surv.*, *53*(6).

Erhan, D., Courville, A. C., Bengio, Y., & Vincent, P. (2010). Why does unsupervised pre-training help deep learning? In Y. W. Teh, & D. M. Titterington (editors), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, Vol. 9 of *JMLR Proceedings* (pp. 201–208). JMLR.org.

Feng, M., & Mazor, B. (1992). Continuous word spotting for applications in telecommunications. In *The Second International Conference on Spoken Language Processing, ICSLP 1992, Banff, Alberta, Canada, October 13-16, 1992*. ISCA.

Fernández-Marqués, J., Tseng, V. W. S., Bhattacharya, S., & Lane, N. D. (2018). Deterministic binary filters for keyword spotting applications. In J. Ott, F. Dressler, S. Saroiu, & P. Dutta (editors), *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2018, Munich, Germany, June 10-15, 2018* (p. 529). ACM.

Gales, M. (1998). Maximum likelihood linear transformations for hmm-based speech recognition. *Computer Speech & Language*, *12*(2): 75–98.

Ge, F., & Yan, Y. (2017). Deep neural network based wake-up-word speech recognition with two-stage detection. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017* (pp. 2761–2765). IEEE.

Ge, W., & Yu, Y. (2017). Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017* (pp. 10–19). IEEE Computer Society.

Gerhard, D. (2003). Pitch extraction and fundamental frequency: History and current techniques. Technical Report TR-CS 2003-06, Department of Computer Science, University of Regina, Regina, Saskatchewan, Canada.

Giannakopoulos, T. (2015). pyaudioanalysis: An open-source python library for audio signal analysis. *PloS one*, *10*(12).

Giraldo, J. S. P., Jain, V., & Verhelst, M. (2021). Efficient execution of temporal convolutional networks for embedded keyword spotting. *IEEE Trans. Very Large Scale Integr. Syst.*, *29*(12): 2220–2228.

Giraldo, J. S. P., & Verhelst, M. (2021). Hardware acceleration for embedded keyword spotting: Tutorial and survey. *ACM Trans. Embed. Comput. Syst.*, *20*(6): 111:1–111:25.

Gish, H., Chow, Y., & Rohlicek, J. R. (1990). Probabilistic vector mapping of noisy speech parameters for HMM word spotting. In *1990 International Conference on Acoustics, Speech, and Signal Processing, ICASSP '90, Albuquerque, New Mexico, USA, April 3-6, 1990* (pp. 117–120). IEEE.

Gish, H., & Ng, K. (1993). A segmental speech model with applications to word spotting. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '93, Minneapolis, Minnesota, USA, April 27-30, 1993* (pp. 447–450). IEEE Computer Society.

Gish, H., Ng, K., & Rohlicek, J. R. (1992). Secondary processing using speech segments for an HMM word spotting system. In *The Second International Conference on Spoken Language Processing, ICSLP 1992, Banff, Alberta, Canada, October 13-16, 1992*. ISCA.

Gruenstein, A., Alvarez, R., Thornton, C., & Ghodrat, M. (2017). A cascade architecture for keyword spotting on mobile devices. *CoRR*, abs/1712.03603.

Guarneri, I., Lauria, A., Farinella, G. M., & Santoro, C. (2022). Tiny neural network pipeline for vocal commands recognition @edge. In A. Paljic, M. Ziat, & K. Bouatouch (editors), *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2022, Vol. 2: HUCAPP, Online Streaming, February 6-8, 2022* (pp. 249–254). SCITEPRESS.

Guo, J., Kumatani, K., Sun, M., Wu, M., Raju, A., Strom, N., & Mandal, A. (2018). Time-delayed bottleneck highway networks using a DFT feature for keyword spotting. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018* (pp. 5489–5493). IEEE.

Hao, J., & Li, X. (2002). Word spotting based ona posterior measure of keyword confidence. *Journal of Computer Science and Technology*, *17*(4): 491–497.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016* (pp. 770–778). IEEE Computer Society.

Heracleous, P., & Shimizu, T. (2003). An efficient keyword spotting technique using a complementary language for filler models training. In *8th European Conference on Speech Communication and Technology, EUROSPEECH 2003 - INTERSPEECH 2003, Geneva, Switzerland, September 1-4, 2003*. ISCA.

Hermansky, H., Hanson, B. A., & Wakita, H. (1985). Perceptually based linear predictive analysis of speech. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '85, Tampa, Florida, USA, March 26-29, 1985* (pp. 509–512). IEEE.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Kingsbury, B., & Sainath, T. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29: 82–97.

Hou, J., Xie, L., & Fu, Z. (2016). Investigating neural network based query-by-example keyword spotting approach for personalized wake-up word detection in mandarin chinese. In *10th International Symposium on Chinese Spoken Language Processing, ISCSLP 2016, Tianjin, China, October 17-20, 2016* (pp. 1–5). IEEE.

Huang, X., Yang, Q., & Liu, S. (2022). Depthwise-separable residual capsule for robust keyword spotting. In B. Þ. Jónsson, C. Gurrin, M. Tran, D. Dang-Nguyen, A. M. Hu, H. T. T. Binh, & B. Huet (editors), *MultiMedia Modeling - 28th International Conference, MMM 2022, Phu Quoc, Vietnam, June 6-10, 2022, Proceedings, Part II*, Vol. 13142 of *Lecture Notes in Computer Science* (pp. 194–204). Springer.

Hwang, K., Lee, M., & Sung, W. (2015). Online keyword spotting with a character-level recurrent neural network. *CoRR*, abs/1512.08903.

Ida, M., & Yamasaki, R. (1998). An evaluation of keyword spotting performance utilizing false alarm rejection based on prosodic information. In *The 5th International Conference on Spoken Language Processing, Incorporating The 7th Australian International Speech Science and Technology Conference, Sydney Convention Centre, Sydney, Australia, 30th November - 4th December 1998*. ISCA.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Vol. 37*, ICML'15 (p. 448–456). JMLR.org.

Ivanovas, E., & Navakauskas, D. (2012). Towards speaker identification system based on dynamic neural network. *Elektronika ir Elektrotechnika*, 18: 69–72.

Jansen, A., & Niyogi, P. (2009a). An experimental evaluation of keyword-filler hidden markov models. Technical Report TR 2009-02, Department of Computer Science, University of Chicago, Chicago, Illinois, United States.

Jansen, A., & Niyogi, P. (2009b). Point process models for spotting keywords in continuous speech. *IEEE Trans. Audio, Speech & Language Processing*, *17*(8): 1457–1470.

Jansen, A., & Niyogi, P. (2009c). Robust keyword spotting with rapidly adapting point process models. In *INTERSPEECH 2009, 10th Annual Conference of the International Speech Communication Association, Brighton, United Kingdom, September 6-10, 2009* (pp. 2767–2770). ISCA.

Jose, C., Mishchenko, Y., Sénéchal, T., Shah, A., Escott, A., & Vitaladevuni, S. N. P. (2020). Accurate detection of wake word start and end using a CNN. In H. Meng, B. Xu, & T. F. Zheng (editors), *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (pp. 3346–3350). ISCA.

Junkawitsch, J., Ruske, G., & Höge, H. (1997). Efficient methods for detecting keywords in continuous speech. In G. Kokkinakis, N. Fakotakis, & E. Dermatas (editors), *Fifth European Conference on Speech Communication and Technology, EUROSPEECH 1997, Rhodes, Greece, September 22-25, 1997*. ISCA.

Kahn, J., Lee, A., & Hannun, A. Y. (2020). Self-training for end-to-end speech recognition. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4–8, 2020* (pp. 7084–7088). IEEE.

Kavya, H. P., & Karjigi, V. (2014). Sensitive keyword spotting for crime analysis. In *2014 IEEE National Conference on Communication, Signal Processing and Networking (NCCSN)* (pp. 1–6). IEEE.

Keshet, J., Grangier, D., & Bengio, S. (2009). Discriminative keyword spotting. *Speech Communication*, *51*(4): 317–329.

Khne, M., Wolff, M., Eichner, M., & Hoffmann, R. (2004). Voice activation using prosodic features. In *INTERSPEECH 2004 - ICSLP, 8th International Conference on Spoken Language Processing, Jeju Island, Korea, October 4-8, 2004*. ISCA.

Kim, K., Gao, C., Graça, R., Kiselev, I., Yoo, H., Delbrück, T., & Liu, S. (2022). A 23μw solar-powered keyword-spotting ASIC with ring-oscillator-based time-domain feature extraction. In *IEEE International Solid-State Circuits Conference, ISSCC 2022, San Francisco, CA, USA, February 20-26, 2022* (pp. 1–3). IEEE.

Kipyatkova, I. S. (2019). LSTM-based language models for very large vocabulary continuous Russian speech recognition system. In A. A. Salah, A. Karpov, & R. Potapova (editors), *Speech and Computer - 21st International Conference, SPECOM 2019, Istanbul, Turkey, August 20-25, 2019, Proceedings*, Vol. 11658 of *Lecture Notes in Computer Science* (pp. 219–226). Springer.

Klemm, H., Class, F., & Kilian, U. (1995). Word- and phrase spotting with syllable-based garbage modelling. In *Fourth European Conference on Speech Communication and Technology, EUROSPEECH 1995, Madrid, Spain, September 18-21, 1995*. ISCA.

Knill, K., Gales, M. J. F., Ragni, A., & Rath, S. P. (2014). Language independent and unsupervised acoustic models for speech recognition and keyword spotting. In H. Li, H. M. Meng, B. Ma, E. Chng, & L. Xie (editors), *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014* (pp. 16–20). ISCA.

Knill, K., & Young, S. J. (1996). Fast implementation methods for viterbi-based word-spotting. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings, ICASSP '96, Atlanta, Georgia, USA, May 7-10, 1996* (pp. 522–525). IEEE Computer Society.

Kosonocky, S. V., & Mammone, R. J. (1995). A continuous density neural tree network word spotting system. In *1995 International Conference on Acoustics, Speech, and Signal Processing, ICASSP '95, Detroit, Michigan, USA, May 08-12, 1995* (pp. 305–308). IEEE Computer Society.

Kumatani, K., Panchapagesan, S., Wu, M., Kim, M., Strom, N., Tiwari, G., & Mandal, A. (2017). Direct modeling of raw audio with DNNS for wake word detection. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2017, Okinawa, Japan, December 16-20, 2017* (pp. 252–257). IEEE.

Kurniawati, E., Celetto, L., Capovilla, N., & George, S. (2012). Personalized voice command systems in multi modal user interface. In *2012 IEEE International Conference on Emerging Signal Processing Applications, ESPA 2012, Las Vegas, NV, USA, January 12-14, 2012* (pp. 45–47). IEEE.

Këpuska, V., & Klein, T. (2009). A novel wake-up-word speech recognition system, wake-up-word recognition task, technology and evaluation. *Nonlinear Analysis: Theory, Methods & Applications*, *71*(12): e2772–e2789.

Laszko, L. (2016). Using formant frequencies to word detection in recorded speech. In M. Ganzha, L. A. Maciaszek, & M. Paprzycki (editors), *Proceedings of the 2016 Federated Conference on Computer Science and Information Systems, FedCSIS 2016, Gdańsk, Poland, September 11-14, 2016*, Vol. 8 of *Annals of Computer Science and Information Systems* (pp. 797–801). IEEE.

Lehtonen, M. (2005). Hierarchical approach for spotting keywords. Technical Report Idiap-RR-41-2005, IDIAP.

Lengerich, C. T., & Hannun, A. Y. (2016). An end-to-end architecture for keyword spotting and voice activity detection. *CoRR*, abs/1611.09405.

Leow, S. J., Lau, T. S., Goh, A., Peh, H. M., Ng, T. K., Siniscalchi, S. M., & Lee, C. (2012). A new confidence measure combining hidden markov models and artificial neural networks of phonemes for effective keyword spotting. In *8th International Symposium on Chinese Spoken Language Processing, ISCSLP 2012, Kowloon Tong, China, December 5-8, 2012* (pp. 112–116). IEEE.

Li, H., Chaudhari, P., Yang, H., Lam, M., Ravichandran, A., Bhotika, R., & Soatto, S. (2020a). Rethinking the hyperparameters for fine-tuning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Li, Q., & Wang, L. (2014). A novel coding scheme for keyword spotting. In *2014 Seventh International Symposium on Computational Intelligence and Design*, Vol. 2 (pp. 379–382). IEEE.

Li, X., Wei, X., & Qin, X. (2020b). Small-footprint keyword spotting with multi-scale temporal convolution. In H. Meng, B. Xu, & T. F. Zheng (editors), *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (pp. 1987–1991). ISCA.

Lin, J., Kilgour, K., Roblek, D., & Sharifi, M. (2020). Training keyword spotters with limited and synthesized speech data. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020* (pp. 7474–7478). IEEE.

Lin, Z. Q., Chung, A. G., & Wong, A. (2018). Edgespeechnets: Highly efficient deep neural networks for speech recognition on the edge. *CoRR*, abs/1810.08559.

Liu, C., Chiu, C., & Chang, H. (2000). Design of vocabulary-independent mandarin keyword spotters. *IEEE Trans. Speech and Audio Processing*, *8*(4): 483–487.

Liu, H., Abhyankar, A., Mishchenko, Y., Sénéchal, T., Fu, G., Kulis, B., Stein, N. D., Shah, A., & Vitaladevuni, S. N. P. (2020). Metadata-aware end-to-end keyword spotting. In H. Meng, B. Xu, & T. F. Zheng (editors), *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (pp. 2282–2286). ISCA.

Lopatka, K., & Bocklet, T. (2020). State sequence pooling training of acoustic models for keyword spotting. In H. Meng, B. Xu, & T. F. Zheng (editors), *Interspeech 2020, 21st*

Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020 (pp. 4338–4342). ISCA.

López-Espejo, I., Tan, Z., & Jensen, J. (2021). A novel loss function and training strategy for noise-robust keyword spotting. *IEEE ACM Trans. Audio Speech Lang. Process.*, 29: 2254–2266.

Manor, E., & Greenberg, S. (2017). Voice trigger system using fuzzy logic. In *2017 International Conference on Circuits, System and Simulation (ICCSS)* (pp. 113–118). IEEE.

Marcus, J. N. (1992). A novel algorithm for HMM word spotting performance evaluation and error analysis. In *1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '92, San Francisco, California, USA, March 23-26, 1992* (pp. 89–92). IEEE Computer Society.

Menon, R., Kamper, H., van der Westhuizen, E., Quinn, J. A., & Niesler, T. (2019). Feature exploration for almost zero-resource asr-free keyword spotting using a multilingual bottleneck extractor and correspondence autoencoders. In G. Kubin, & Z. Kacic (editors), *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019* (pp. 3475–3479). ISCA.

Mo, T., Yu, Y., Salameh, M., Niu, D., & Jui, S. (2020). Neural architecture search for keyword spotting. In H. Meng, B. Xu, & T. F. Zheng (editors), *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (pp. 1982–1986). ISCA.

Morgan, D. P., & Scofield, C. L. (1991). *Neural Networks and Speech Processing* (pp. 329–348). Springer US, Boston, MA.

Morgan, D. P., Scofield, C. L., & Adcock, J. E. (1991). Multiple neural network topologies applied to keyword spotting. In *1991 International Conference on Acoustics, Speech, and Signal Processing, ICASSP '91, Toronto, Ontario, Canada, May 14-17, 1991* (pp. 313–316). IEEE Computer Society.

Morgan, D. P., Scofield, C. L., Lorenzo, T. M., Real, E. C., & Loconto, D. P. (1990). A keyword spotter which incorporates neural networks for secondary processing. In *1990 International Conference on Acoustics, Speech, and Signal Processing, ICASSP '90, Albuquerque, New Mexico, USA, April 3-6, 1990* (pp. 113–116). IEEE.

Mussakhojayeva, S., Khassanov, Y., & Varol, H. A. (2021). A study of multilingual end-to-end speech recognition for Kazakh, Russian, and English. In A. Karpov, & R. Potapova (editors), *Speech and Computer — 23rd International Conference, SPECOM 2021, St. Petersburg, Russia, September 27-30, 2021, Proceedings*, Vol. 12997 of *Lecture Notes in Computer Science* (pp. 448–459). Springer.

Myer, S., & Tomar, V. S. (2018). Efficient keyword spotting using time delay neural networks. In B. Yegnanarayana (editor), *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018.* (pp. 1264–1268). ISCA.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In J. Fürnkranz, & T. Joachims (editors), *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel* (pp. 807–814). Omnipress.

Naylor, J. A., Huang, W. Y., Nguyen, M., & Li, K. P. (1992). The application of neural networks to wordspotting. In *Conference Record of the Twenty-Sixth Asilomar Conference on Signals, Systems & Computers* (pp. 1081–1085). Los Alamitos, CA, USA, IEEE Computer Society.

Ng, D., Chen, Y., Tian, B., Fu, Q., & Chng, E. S. (2022a). Convmixer: Feature interactive convolution with curriculum learning for small footprint and noisy far-field keyword spotting. *CoRR*, abs/2201.05863.

Ng, D., Pang, J. H., Xiao, Y., Tian, B., Fu, Q., & Chng, E. S. (2022b). Small footprint multi-channel convmixer for keyword spotting with centroid based awareness. *CoRR*, abs/2204.05445.

Panayotov, V., Chen, G., Povey, D., & Khudanpur, S. (2015). Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015* (pp. 5206–5210). IEEE.

Park, D. S., Chan, W., Zhang, Y., Chiu, C., Zoph, B., Cubuk, E. D., & Le, Q. V. (2019). SpecAugment: A simple data augmentation method for automatic speech recognition. In G. Kubin, & Z. Kacic (editors), *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019* (pp. 2613–2617). ISCA.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32* (pp. 8024–8035). Curran Associates, Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830.

Pipiras, L., Maskeliūnas, R., & Damaševičius, R. (2019). Lithuanian speech recognition using purely phonetic deep learning. *Computers*, *8*(4).

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., & Vesely, K. (2011). The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

Rasymas, T., & Rudzionis, V. (2014). Combining multiple foreign language speech recognizers by using neural networks. In A. Utka, G. Grigonyte, J. Kapociute-Dzikiene, & J.

Vaicenoniene (editors), *Human Language Technologies - The Baltic Perspective - Proceedings of the Sixth International Conference Baltic HLT 2014, Kaunas, Lithuania, September 26-27, 2014*, Vol. 268 of *Frontiers in Artificial Intelligence and Applications* (pp. 33–39). IOS Press.

Ravanelli, M., & Bengio, Y. (2018). Speaker recognition from raw waveform with sincnet. In *2018 IEEE Spoken Language Technology Workshop, SLT 2018, Athens, Greece, December 18-21, 2018* (pp. 1021–1028). IEEE.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (editors), *Advances in Neural Information Processing Systems*, Vol. 28. Curran Associates, Inc.

Rogina, I., & Waibel, A. (1995). *Janus toolkit documentation*. http://www.cs.cmu.edu/~tanja/Lectures/JRTkDoc/OldDoc/senones/sn_main.html (accessed on 19 May 2021).

Rohlicek, J. R., Jeanrenaud, P., Ng, K., Gish, H., Musicus, B. R., & Siu, M. (1993). Phonetic training and language modeling for word spotting. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '93, Minneapolis, Minnesota, USA, April 27-30, 1993* (pp. 459–462). IEEE Computer Society.

Rohlicek, J. R., Russell, W., Roukos, S., & Gish, H. (1989). Continuous hidden Markov modeling for speaker-independent word spotting. In *International Conference on Acoustics, Speech, and Signal Processing,* (pp. 627–630). IEEE.

Rose, R. C., & Paul, D. B. (1990). A hidden markov model based keyword recognition system. In *1990 International Conference on Acoustics, Speech, and Signal Processing, ICASSP '90, Albuquerque, New Mexico, USA, April 3-6, 1990* (pp. 129–132). IEEE.

Rudzionis, A., & Rudzionis, V. (2002). Lithuanian speech database LTDIGITS. In *Proceedings of the Third International Conference on Language Resources and Evaluation, LREC 2002, May 29-31, 2002, Las Palmas, Canary Islands, Spain*. European Language Resources Association.

Rybakov, O., Kononenko, N., Subrahmanya, N., Visontai, M., & Laurenzo, S. (2020). Streaming keyword spotting on mobile devices. In H. Meng, B. Xu, & T. F. Zheng (editors), *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (pp. 2277–2281). ISCA.

Sadhu, S., & Ghosh, P. K. (2017). Low resource point process models for keyword spotting using unsupervised online learning. In *25th European Signal Processing Conference, EUSIPCO 2017, Kos, Greece, August 28 - September 2, 2017* (pp. 538–542). IEEE.

Sainath, T. N., & Parada, C. (2015). Convolutional neural networks for small-footprint keyword spotting. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015* (pp. 1478–1482). ISCA.

Sainath, T. N., Weiss, R. J., Senior, A. W., Wilson, K. W., & Vinyals, O. (2015). Learning the speech front-end with raw waveform cldnns. In *INTERSPEECH 2015, 16th Annual*

Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015 (pp. 1–5). ISCA.

Salimbajevs, A., & Kapociute-Dzikiene, J. (2018). General-purpose lithuanian automatic speech recognition system. In K. Muischnek, & K. Müürisep (editors), *Human Language Technologies - The Baltic Perspective - Proceedings of the Eighth International Conference Baltic HLT 2018, Tartu, Estonia, 27-29 September 2018*, Vol. 307 of *Frontiers in Artificial Intelligence and Applications* (pp. 150–157). IOS Press.

Sangeetha, J., & Jothilakshmi, S. (2014). A novel spoken keyword spotting system using support vector machine. *Eng. Appl. of AI*, 36: 287–293.

Schneider, S., Baevski, A., Collobert, R., & Auli, M. (2019). wav2vec: Unsupervised pre-training for speech recognition. In G. Kubin, & Z. Kacic (editors), *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019* (pp. 3465–3469). ISCA.

Seo, D., Oh, H.-S., & Jung, Y. (2021). Wav2KWS: Transfer learning from speech representations for keyword spotting. *IEEE Access*, 9: 80682–80691.

Shan, C., Zhang, J., Wang, Y., & Xie, L. (2018). Attention-based end-to-end models for small-footprint keyword spotting. In B. Yegnanarayana (editor), *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018* (pp. 2037–2041). ISCA.

Shokri, A., Davarpour, M. H., & Akbari, A. (2014). Improving keyword detection rate using a set of rules to merge hmm-based and svm-based keyword spotting results. In *2014 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2014, Delhi, India, September 24-27, 2014* (pp. 1715–1718). IEEE.

Shokri, A., Davarpour, M. H., Akbari, A., & Nasersharif, B. (2013). Detecting keywords in persian conversational telephony speech using a discriminative english keyword spotter. In *IEEE International Symposium on Signal Processing and Information Technology, Athens, Greece, December 12-15, 2013* (pp. 272–276). IEEE Computer Society.

Shokri, A., Tabibian, S., Akbari, A., Nasersharif, B., & Kabudian, J. (2011). A robust keyword spotting system for Persian conversational telephone speech using feature and score normalization and ARMA filter. In *2011 IEEE GCC Conference and Exhibition (GCC)* (pp. 497–500). IEEE.

Siegert, I., Weißkirchen, N., Krüger, J., Akhtiamov, O., & Wendemuth, A. (2021). Admitting the addressee detection faultiness of voice assistants to improve the activation performance using a continuous learning framework. *Cogn. Syst. Res.*, 70: 65–79.

Silaghi, M., & Vargiya, R. (2005). A new evaluation criteria for keyword spotting techniques and a new algorithm. In *INTERSPEECH 2005 - Eurospeech, 9th European Conference on Speech Communication and Technology, Lisbon, Portugal, September 4-8, 2005* (pp. 1593–1596). ISCA.

Siu, M., Gish, H., & Rohlicek, J. R. (1994). Predicting word spotting performance. In *The 3rd International Conference on Spoken Language Processing, ICSLP 1994, Yokohama, Japan, September 18-22, 1994*. ISCA.

Smirnov, V., Ignatov, D., Gusev, M., Farkhadov, M. P., Rumyantseva, N., & Farkhadova, M. (2016). A Russian keyword spotting system based on large vocabulary continuous speech recognition and linguistic knowledge. *Journal of Electrical and Computer Engineering*, 2016: 1–9.

Sun, M., Snyder, D., Gao, Y., Nagaraja, V., Rodehorst, M., Panchapagesan, S., Strom, N., Matsoukas, S., & Vitaladevuni, S. (2017). Compressed time delay neural network for small-footprint keyword spotting. In F. Lacerda (editor), *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017* (pp. 3607–3611). ISCA.

Szöke, I., Grézl, F., Cernocký, J., Fapso, M., & Cipr, T. (2010). Acoustic keyword spotter - optimization from end-user perspective. In D. Hakkani-Tür, & M. Ostendorf (editors), *2010 IEEE Spoken Language Technology Workshop, SLT 2010, Berkeley, California, USA, December 12-15, 2010* (pp. 189–193). IEEE.

Szöke, I., Schwarz, P., Matejka, P., Burget, L., Karafiát, M., & Cernocký, J. (2005). Phoneme based acoustics keyword spotting in informal continuous speech. In V. Matousek, P. Mautner, & T. Pavelka (editors), *Text, Speech and Dialogue, 8th International Conference, TSD 2005, Karlovy Vary, Czech Republic, September 12-15, 2005, Proceedings*, Vol. 3658 of *Lecture Notes in Computer Science* (pp. 302–309). Springer.

Szöke, I., Skácel, M., Burget, L., & Cernocký, J. (2015). Coping with channel mismatch in query-by-example - but QUESST 2014. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015* (pp. 5838–5842). IEEE.

Tabibian, S. (2017). A voice command detection system for aerospace applications. *International Journal of Speech Technology*, *20*(4): 1049–1061.

Tabibian, S., Akbari, A., & Nasersharif, B. (2011). An evolutionary based discriminative system for keyword spotting. In *2011 International Symposium on Artificial Intelligence and Signal Processing (AISP)* (pp. 83–88). IEEE.

Tabibian, S., Akbari, A., & Nasersharif, B. (2013). Keyword spotting using an evolutionary-based classifier and discriminative features. *Eng. Appl. of AI*, *26*(7): 1660–1670.

Tabibian, S., Akbari, A., & Nasersharif, B. (2014). Extension of a kernel-based classifier for discriminative spoken keyword spotting. *Neural Processing Letters*, *39*(2): 195–218.

Tabibian, S., Akbari, A., & Nasersharif, B. (2016). A fast hierarchical search algorithm for discriminative keyword spotting. *Inf. Sci.*, 336: 45–59.

Tabibian, S., Akbari, A., & Nasersharif, B. (2018). Discriminative keyword spotting using triphones information and n-best search. *Inf. Sci.*, 423: 157–171.

Tang, R., & Lin, J. (2018). Deep residual learning for small-footprint keyword spotting. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018* (pp. 5484–5488). IEEE.

TensorFlow Core Team (2021). *Simple audio recognition: Recognizing keywords*. https://www.tensorflow.org/tutorials/audio/simple_audio (accessed on 13 December 2021).

Tetariy, E., Bar-Yosef, Y., Silber-Varod, V., Gishri, M., Alon-Lavi, R., Aharonson, V., Opher, I., & Moyal, A. (2015). Cross-language phoneme mapping for phonetic search keyword spotting in continuous speech of under-resourced languages. *Artif. Intell. Res.*, 4: 72–82.

Triguero, I., García, S., & Herrera, F. (2015). Self-labeled techniques for semi-supervised learning: Taxonomy, software and empirical study. *Knowledge and Information Systems*, 42.

Ulkar, M. G., & Okman, O. E. (2021). Ultra-low power keyword spotting at the edge. *CoRR*, abs/2111.04988.

Vasilache, M., & Vasilache, A. (2009). Keyword spotting with duration constrained HMMs. In *17th European Signal Processing Conference, EUSIPCO 2009, Glasgow, Scotland, UK, August 24-28, 2009* (pp. 1230–1234). IEEE.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., & SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272.

Vroomen, L. C., & Normandin, Y. (1992). Robust speaker-independent hidden Markov model based word spotter. In P. Laface, & R. De Mori (editors), *Speech Recognition and Understanding* (pp. 95–100). Berlin, Heidelberg, Springer Berlin Heidelberg.

Wang, H., Ragni, A., Gales, M. J. F., Knill, K. M., Woodland, P. C., & Zhang, C. (2015). Joint decoding of tandem and hybrid systems for improved keyword spotting on low resource languages. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015* (pp. 3660–3664). ISCA.

Warden, P. (2018). Speech commands: A dataset for limited-vocabulary speech recognition. *CoRR*, abs/1804.03209.

Wikipedia contributors (2021). *Syllable — Wikipedia, the free encyclopedia*. https://en.wikipedia.org/w/index.php?title=Syllable&oldid=1006686720.

Wikipedia contributors (2021). *Phoneme — Wikipedia, the free encyclopedia*. https://en.wikipedia.org/w/index.php?title=Phoneme&oldid=1006518035.

Wilcox, L. D., & Bush, M. A. (1992). Training and search algorithms for an interactive wordspotting system. In *1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '92, San Francisco, California, USA, March 23-26, 1992* (pp. 97–100). IEEE Computer Society.

Wöllmer, M., Eyben, F., Graves, A., Schuller, B. W., & Rigoll, G. (2009a). Improving keyword spotting with a tandem BLSTM-DBN architecture. In J. S. Casals, & V. Zaiats (editors), *Advances in Nonlinear Speech Processing, International Conference on Nonlinear Speech Processing, NOLISP 2009, Vic, Spain, June 25-27. Revised Selected Papers*, Vol. 5933 of *Lecture Notes in Computer Science* (pp. 68–75). Springer.

Wöllmer, M., Eyben, F., Keshet, J., Graves, A., Schuller, B. W., & Rigoll, G. (2009b). Robust discriminative keyword spotting for emotionally colored spontaneous speech using bidirectional LSTM networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2009, 19-24 April 2009, Taipei, Taiwan* (pp. 3949–3952). IEEE.

Wöllmer, M., Schuller, B. W., & Rigoll, G. (2013). Keyword spotting exploiting long short-term memory. *Speech Communication*, *55*(2): 252–265.

Wu, H., Jia, Y., Nie, Y., & Li, M. (2020). Domain aware training for far-field small-footprint keyword spotting. In H. Meng, B. Xu, & T. F. Zheng (editors), *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (pp. 2562–2566). ISCA.

Wu, M., Panchapagesan, S., Sun, M., Gu, J., Thomas, R., Vitaladevuni, S. N. P., Hoffmeister, B., & Mandal, A. (2018). Monophone-based background modeling for two-stage on-device wake word detection. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018* (pp. 5494–5498). IEEE.

Xu, M., & Zhang, X. (2020). Depthwise separable convolutional resnet with squeeze-and-excitation blocks for small-footprint keyword spotting. In H. Meng, B. Xu, & T. F. Zheng (editors), *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (pp. 2547–2551). ISCA.

Xu, Q., Likhomanenko, T., Kahn, J., Hannun, A., Synnaeve, G., & Collobert, R. (2020). Iterative pseudo-labeling for speech recognition. In H. Meng, B. Xu, & T. F. Zheng (editors), *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (pp. 1006–1010). ISCA.

Yang, C., Wen, X., & Song, L. (2020). Multi-scale convolution for robust keyword spotting. In H. Meng, B. Xu, & T. F. Zheng (editors), *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (pp. 2577–2581). ISCA.

Yilmaz, E., Gevrek, Ö. B., Wu, J., Chen, Y., Meng, X., & Li, H. (2020). Deep convolutional spiking neural networks for keyword spotting. In H. Meng, B. Xu, & T. F. Zheng (editors), *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (pp. 2557–2561). ISCA.

Yu, D., & Deng, L. (2014). *Automatic Speech Recognition: A Deep Learning Approach*. Springer Publishing Company, Incorporated.

Zeghidour, N., Usunier, N., Synnaeve, G., Collobert, R., & Dupoux, E. (2018). End-to-end speech recognition from the raw waveform. In B. Yegnanarayana (editor), *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018* (pp. 781–785). ISCA.

Zehetner, A., Hagmüller, M., & Pernkopf, F. (2014). Wake-up-word spotting for mobile systems. In *22nd European Signal Processing Conference, EUSIPCO 2014, Lisbon, Portugal, September 1-5, 2014* (pp. 1472–1476). IEEE.

Zeppenfeld, T., & Waibel, A. H. (1992). A hybrid neural network, dynamic programming word spotter. In *1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '92, San Francisco, California, USA, March 23-26, 1992* (pp. 77–80). IEEE Computer Society.

Zhang, K., Wu, Z., Yuan, D., Luan, J., Jia, J., Meng, H., & Song, B. (2020). Re-weighted interval loss for handling data imbalance problem of end-to-end keyword spotting. In H. Meng, B. Xu, & T. F. Zheng (editors), *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (pp. 2567–2571). ISCA.

Zhang, P., & Zhang, X. (2020). Deep template matching for small-footprint and configurable keyword spotting. In H. Meng, B. Xu, & T. F. Zheng (editors), *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (pp. 2572–2576). ISCA.

Zhang, S., Liu, W., & Qin, Y. (2016). Wake-up-word spotting using end-to-end deep neural network system. In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016* (pp. 2878–2883). IEEE.

Zhang, X., Su, Z., & Rekimoto, J. (2022). Aware: Intuitive device activation using prosody for natural voice interactions. In S. D. J. Barbosa, C. Lampe, C. Appert, D. A. Shamma, S. M. Drucker, J. R. Williamson, & K. Yatani (editors), *CHI '22: CHI Conference on Human Factors in Computing Systems, New Orleans, LA, USA, 29 April 2022 - 5 May 2022* (pp. 432:1–432:16). ACM.

Zheng, F., Xu, M., Mou, X., Wu, J., Wu, W., & Fang, D. (1999). Harkman - A vocabulary-independent keyword spotter for spontaneous chinese speech. *J. Comput. Sci. Technol.*, *14*(1): 18–26.

Zhu, C., Kong, Q., Zhou, L., Xiong, G., & Zhu, F. (2013). Sensitive keyword spotting for voice alarm systems. In *Proceedings of 2013 IEEE International Conference on Service Operations and Logistics, and Informatics* (pp. 350–353). IEEE.

List of Scientific Publications by the Author on the Topic of the Dissertation

Papers in the Reviewed Scientific Journals

Kolesau, A., & Šešok, D. (2021b). Voice activation for low-resource languages. *Applied Sciences*, *11*(14): 1–16. https://doi.org/10.3390/app11146298

Kolesau, A., & Šešok, D. (2020b). Unsupervised pre-training for voice activation. *Applied Sciences*, *10*(23): 1–13. https://doi.org/10.3390/app10238643

Kolesau, A., & Šešok, D. (2020c). Voice activation systems for embedded devices: Systematic literature review. *Informatica*, *31*(1): 65–88. https://doi.org/10.15388/20-INFOR398

Other Papers

Kolesau, A., & Šešok, D. (2021). On the acoustic unit choice for the keyword spotting problem. In *VI International scientific conference "High Technologies. Business. Society"*, Vol. 6 (pp. 5–7). Sofia: Scientific Technical Union Of Mechanical Engineering Industry-4.0.

Kolesau, A., & Šešok, D. (2021a). Detecting wake-word repeats in voice-activated systems. In *2021 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)* (pp. 1–4). IEEE. https://doi.org/10.1109/eStream53087.2021.9431481

Kolesau, A., & Šešok, D. (2020a). Investigation of acoustic features for voice activation problem. In *2020 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)* (pp. 1–4). IEEE. https://doi.org/10.1109/eStream50540.2020.9108867

Summary in Lithuanian

Įvadas

Problemos formulavimas

Iš anksto nustatyto reikšminio žodžio (arba frazės) interpretuojamą paieškos algoritmą yra sudėtinga suformuluoti. Todėl šiam uždaviniui spręsti plačiai taikomi mašininio mokymosi metodai, ypač giliojo mokymosi metodai, kurie leidžia išgauti iš duomenų dėsningumus vietoje šių dėsningumų formulavimo ekspertų pagalba.

Vienas iš tokio sprendimo būdo trūkumų yra būtinybė turėti labai didelius duomenų rinkinius. Pavyzdžiui, Chen et al. (2014a) savo straipsnyje naudoja šimtus tūkstančių garsinių pavyzdžių reikšminiam žodžiui, o Jose et al. (2020) naudoja milijonus. Maža to, anglų kalbai dar egzistuoja keli atviri dideli duomenų rinkiniai (visų pirma, *Google Speech Commands* (Warden, 2018)), o kitoms kalboms tokių duomenų rinkinių beveik nėra. Todėl sunku patikrinti, ar atrasti geriausi balso aktyvavimo metodai anglų kalbai gerai tinka ir kitoms kalboms.

Be to, dauguma tyrėjų, dirbančių su reikšminio žodžio paieškos (*keyword spotting*) uždaviniais, susikoncentruoja ties tos sistemos dalies gerinimu, kurią sudaro neuroninis tinklas. Tačiau, galima parodyti, kad ir kitos sistemos dalys turi svarbią įtaką kokybei.

Darbo aktualumas

Tyrėjai, kurie dirba su reikšminio žodžio aktyvavimo užduotimi, dažnai taiko sudėtingus mašininio mokymosi metodus. Dažniausiai efektyviam tokių metodų darbui reikalingi dideli duomenų rinkiniai, o surinkti didelį duomenų rinkinį paprastai yra sudėtinga. Todėl daugelis

tyrėjų tikrina savo rezultatus su kalbomis, kurioms tradiciškai yra daug duomenų, visų pirma tai yra anglų kalba.

Populiariausias duomenų rinkinys reikšminio žodžio aktyvavimo uždaviniui šiuo metu yra *Google Speech Commands* (Warden, 2018). Be abejonių, tokio duomenų rinkinio buvimas leidžia vystyti balso aktyvavimo tematiką, tačiau tokia situacija, deja, daro tyrimus kažkuria prasme vienašališkus.

Pagrindinis tyrimų akcentas šiuo momentu yra gerinimas tos sistemos dalies, kuri realizuota naudojant neuroninį tinklą, nes dažniausiai būtent ši dalis duoda didžiausią kokybės padidėjimą, ypač jeigu duomenų rinkinys mokymuisi yra pakankamai didelis. Be to, didžiausia hipotezių dalis net netikrinama kalboms su nedideliu kiekiu paruoštų duomenų, pavyzdžiui, lietuvių kalbai.

Duomenų rinkinio konstravimas ir metodų, kurie duoda geriausius rezultatus kalboms su nedideliu kiekiu paruoštų duomenų, tyrimas, leistų nagrinėti ir kurti balso aktyvavimo sistemas tokioms kalboms.

Tyrimų objektas

Šio darbo tyrimo objektas yra giliojo mokymosi metodų taikymas balso aktyvavimo sistemose, ypač tose sistemos dalyse, kurios nesusijusios su neuroniniu tinklu, ir darbui su kalbomis, neturinčiomis didelių kiekių paruoštų duomenų.

Darbo tikslas

Šio darbo tikslas yra balso aktyvavimo sistemų kokybės gerinimas, modifikuojant tas sistemos dalis, kurios nesusijusios su neuroniniu tinklu ir skirtos kalboms su nedideliu kiekiu paruoštų duomenų.

Darbo uždaviniai

Tikslui pasiekti iškelti šie uždaviniai:

1. Atlikti mokslinės literatūros analizę, siekiant nustatyti esamą situaciją balso aktyvavimo sistemų tyrimuose.
2. Įvertinti audiospektrogramos panaudojimo galimybę kaip garso požymių išgavimo metodą, atsižvelgiant į išgaunamų garso požymių supaprastinimo tendenciją, ir išanalizuoti hiperparametrus, naudojamus nagrinėjamuose metoduose.
3. Pasiūlyti ir įvertinti naujus akustinius vienetus, kurie reikalauja mažesnių *a priori* žinių apie balso aktyvavimo sistemų kalbą, ir palyginti juos su esamais.
4. Pasiūlyti ir įvertinti fonemomis pagrįstų balso aktyvinimo sistemų dekodavimo modulio modifikaciją, siekiant pagerinti reikšminių žodžių pasikartojimų aptikimą 10 %.
5. Surinkti lietuviškų balso komandų duomenų rinkinį vertinimui ir tolesniems tyrimams apie balso aktyvavimo sistemas, esant sąlygoms, kai yra mažiau nei 20 mokymo pavyzdžių kiekvienam reikšminiam žodžiui.

6. Pasiūlyti ir įvertinti giliojo mokymosi metodus, siekiant pagerinti aptikimo tikslumą 7 % balso aktyvavimo sistemai lietuvių kalbai, turint mažiau nei 20 mokymo pavyzdžių vienam reikšminiam žodžiui.

Tyrimų metodika

Šiame darbe buvo taikomi kiekybiniai tyrimų metodai. Tam, kad gautume geresnį suvokimą apie dabartinę balso aktyvavimo sistemų tyrimų būklę bei apie šių sistemų vystymo galimybes, buvo atlikta mokslinės literatūros analizė. Keli eksperimentai buvo atlikti siekiant sukurti naują duomenų rinkinį balso aktyvavimo sistemų, skirtų kalbų su turimu nedideliu kiekiu paruoštų duomenų tyrimui ir geresnių mašininio bei giliojo mokymosi metodų nustatymui. Duomenų rinkiniui konstruoti ir keliems skaitiniams eksperimentams atlikti buvo naudojamas kompiuteris su procesoriumi Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz, 8 GB operatyviosios atminties, Ubuntu 18.04 operacine sistema ir programine įranga Audacity. Kitas kompiuteris su procesoriumi Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz, 32 GB operatyvinės atminties, Tesla M40 24GB GPU grafine korta, operacine sistema Ubuntu 18.08 ir bibliotekomis Keras, Tensorflow, PyTorch bei kaldi buvo naudojamas visiems likusiems eksperimentams atlikti. Gauti duomenys buvo analizuojami naudojant tipines metrikas, skirtas balso aktyvavimo sistemoms: tikslumas (*precision*), pilnumas (*recall*), klaidingų suveikimų dalis (*false accept rate*) ir klaidingų nesuveikimų dalis (*false reject rate*).

Darbo mokslinis naujumas

Šio darbo mokslinį naujumą sudaro:

- Garso požymių išgavimo metodo hiperparametrų tyrimas parodė, kad 40 ms ir 55 ms kadro ilgio reikšmės mel dažnio cepstralių koeficientų (*mel frequency cepstral coefficients*) metodui leidžia 0,95 % ± 0,77 % (95 % pasikliautinasis intervalas) pagerinti aptikimo kokybę, lyginant su pagal numatytuosius parametrus 20 ms ir 30 ms vertėmis.
- Aptikimo tikslumas yra 11,78 % ± 7,77 % (95 % pasikliautinasis intervalas) didesnis, kai visi treniruočių rinkinyje esantys akustinių vienetų pavyzdžiai naudojami kaip mokymo tikslai, palyginus su tuo atveju, kai naudojami tik tie akustinių vienetų pavyzdžiai, kurie yra reikšminio žodžio dalis.
- Buvo pasiūlytas esamo dekodavimo modulio modifikavimo metodas, kuris leidžia aptikti 13,32 % ± 1,76 % (95 % pasikliautinasis intervalas) anksčiau praleistų reikšminio žodžio pasikartojimų be klaidingų aktyvavimų padidėjimo.
- Buvo sukurtas naujas duomenų rinkinys lietuvių kalbai. Rinkinys gali būti naudojamas balso sistemoms konstruoti ir jų kokybei vertinti, kai dirbama su nedideliais duomenų rinkiniais. Rinkinį sudaro 20 reikšminių žodžių įrašai, kuriuos tarė 28 savanoriai naudodami mobilųjį telefoną, ir 292 įrašai be kalbos.
- Buvo pasiūlyti ir įvertinti balso aktyvavimo sistemų konstravimo metodai, skirti darbui su nedideliais duomenų rinkiniais. Geriausias iš pasiūlytų metodų palyginus

su standartiniu baziniu sprendiniu parodė santykinį pagerėjimą 10,17 % ± 2,11 % (95 % pasikliautinasis intervalas) tiesioginio pasiskirstymo ir liekamųjų neuroninių tinklų architektūroms.

Darbo rezultatų praktinė reikšmė

Naujas duomenų rinkinys lietuvių kalbai leis mokslininkams tęsti tyrimus konstruojant efektyvias balso aktyvavimo sistemas, skirtas kalboms su nedideliu kiekiu paruoštų duomenų. Taip pat tyrėjai galės patikrinti savo išvadas tokioms kalboms.

Pasiūlymai modifikuoti tas balso aktyvavimo sistemų dalis, kurios nėra neuroninis tinklas, atveria naujus tokių sistemų derinimo būdus, dažniausiai nereikalausiančius sudėtingo pakartotinio neuroninio tinklo apmokymo.

Be to, tai pabrėžia gilesnio šios balso aktyvavimo srities tyrimo svarbą. Taip pat pasiūlyti giliojo apmokymo metodai balso aktyvavimo sistemų, skirtų kalboms su nedideliu kiekiu paruoštų duomenų konstruoti, leidžia tokių kalbų atveju konstruoti kokybiškesnes sistemas.

Ginamieji teiginiai

Darbo rezultatų pagrindu buvo suformuluoti toliau pateikti ginamieji teiginiai:

1. Kadro ilgio reikšmės 40 ms ir 55 ms mel dažnio cepstralių koeficientų (*mel frequency cepstral coefficients*) metodui, *Google Speech Commands* duomenų rinkiniui (Warden, 2018) ir konvoliuciniams neuroniniams tinklams leidžia 0,95 % ± 0,77 % (95 % pasikliautinasis intervalas) pagerinti aptikimo kokybę, lyginant su pagal numatytuosius parametrus nustatytomis 20 ms ir 30 ms vertėmis.

2. Aptikimo tikslumas yra 11,78 % ± 7,77 % (95 % pasikliautinasis intervalas) didesnis, kai visi treniruočių rinkinyje esantys akustinių vienetų pavyzdžiai naudojami kaip mokymo tikslai, palyginus su tuo atveju, kai naudojami tik tie akustinių vienetų pavyzdžiai, kurie yra reikšminio žodžio dalis.

3. Pasiūlytas balso aktyvavimo sistemų, pagrįstų foneminiais akustiniais vienetais, dekodavimo modulio modifikavimas leidžia aptikti 13,32 % ± 1,76 % (95 % pasikliautinasis intervalas) anksčiau praleistų reikšminio žodžio pakartojimų be naujų klaidingų aktyvavimų. Leidus tik 1% naujų klaidingų aktyvavimų, galima rasti 79,9 % ± 1,5 % (95 % pasikliautinasis intervalas) anksčiau praleistų pakartojimų. Siūlomas metodas nereikalauja pakartotinio neuroninio tinklo apmokymo ir padidina skaičiavimų laiką tokio pat ilgio garso įrašui blogiausiu atveju 4 %.

4. Pasiūlytas išankstinis garso požymių išgavimo modulio apmokymas taikant neprižiūrimą išankstinį mokymą (*unsupervised pre-training*) pagerina balso aktyvavimo tikslumą (*accuracy*) 8,04 % ± 4,73 % (95 % pasikliautinasis intervalas), kai kiekvieno žodžio mokymo pavyzdžių skaičius neviršija 7, ir 24,68 % ± 6,24 % (95 % pasikliautinasis intervalas), kai yra ne daugiau kaip 5 mokymo pavyzdžiai lietuvių, anglų ir rusų kalboms.

5. Siūlomas bendro apmokymo metodas lietuviškame mokymo rinkinyje ir anglų kalbos mokymo rinkinyje rodo santykinį tikslumo pagerėjimą 10,17 % ± 2,11 %

(95 % pasikliautinis intervalas) tiesioginio pasiskirstymo ir liekamųjų neuroninių tinklų architektūroms naujam lietuviškam duomenų rinkiniui ir *Google Speech Commands* duomenų rinkiniui (Warden, 2018).

Darbo rezultatų aprobavimas

Šio darbo rezultatai buvo publikuoti 6 mokslo straipsniuose, iš kurių 3 buvo publikuoti *Clarivate Analytics* (taip pat žinomas kaip *Thomson Reuters*) *Web of Science* žurnaluose. 3 straipsniai buvo publikuoti konferencijų medžiagoje. Autorius padarė 4 pranešimus Lietuvos ir tarptautinėse konferencijose:

- *The 11th International Workshop on Data Analysis Methods for Software Systems (DAMSS 2019)*, 2019, Druskininkai, Lietuva.
- *2020 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*, 2020, Vilnius, Lietuva.
- *VI International scientific conference „High Technologies. Business. Society“*, 2021, Borovets, Bulgarija.
- *2021 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*, 2021, Vilnius, Lietuva.

Disertacijos struktūra

Šį mokslinį darbą sudaro įvadas, 3 pagrindiniai skyriai, bendrosios išvados, literatūros sąrašas, autoriaus publikacijų sąrašas ir priedai.

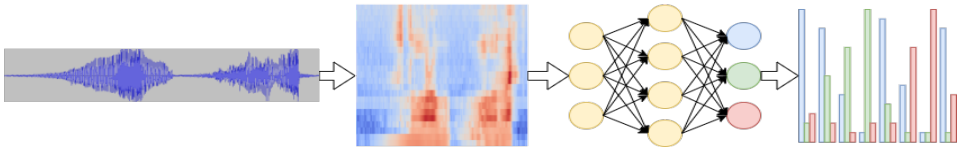
Bendra disertacijos apimtis – 121 puslapiai be priedų.

Darbe yra 21 lygtys, 32 iliustracijos, 51 lentelės. Disertacijos tekste cituojami 164 darbai.

1. Literatūros apžvalga balso aktyvavimo sistemų tematika

Šiame skyriuje atlikta balso aktyvavimo sistemų konstravimo metodų apžvalga. Aprašomi tokių sistemų ypatumai bei paskirtis, taip pat nagrinėjamos pagrindinės balso aktyvavimo sistemų charakteristikos. Be to, šiame skyriuje siūloma tipinės balso aktyvavimo sistemos struktūra, kurią naudojant patogu lyginti skirtingus požiūrius. Aprašomi skirtingi kiekvieno šios struktūros modulio konstravimo požiūriai: garso požymių ištraukimo modulis (*acoustic feature extraction pipeline*), akustinis modelis (*acoustic model*), dekodavimo modulis (*decoding*). Siūloma struktūra pavaizduota S1.1 pav.

Balso aktyvavimo uždavinys yra glaudžiai susijęs su automatiniu kalbos atpažinimo uždaviniu (*automatic speech recognition*) ir ištarto termino aptikimo (*spoken term detection*) uždaviniu. Kalbos atpažinimo uždavinys – surasti labiausiai tikėtiną žodžių seką, kuri buvo pasakyta per garso įrašą. Balso aktyvavimo uždavinyje reikia surasti iš anksto nustatytą žodžių arba frazių rinkinį arba nustatyti, kad garso sraute tokių žodžių / frazių nebuvo ištarta. Natūralu, kad kalbos atpažinimo uždavinys yra bendresnis, tačiau praktikoje šis uždavinys



S1.1 pav. Tipinės balso aktyvavimo sistemos struktūra

reikalauja didelių skaičiavimo resursų, tai daro balso aktyvavimo sistemas, sudarytas kalbos atpažinimo pagrindu, per brangias reikšminiam naudojimui.

Ištarto žodžio aptikimo uždavinys – tai duotos frazės paieškos uždavinys (ir frazė gali keistis nuo užklausos iki užklausos) fiksuotame garsinių duomenų rinkinyje. Priešingai, balso aktyvavimo uždavinyje reikšminis žodis arba frazė yra fiksuoti, o garsiniai duomenys teikiami realiojo laiko režimu. Todėl ištarto termino aptikimo uždavinyje galima naudoti vadinamuosius neprisijungus požiūrius, t. y. požiūrius, kurie turi atsitiktinį priėjimą (*random access*) prie visų garso duomenų, pavyzdžiui, dvikrypčiai rekurentiniai neuroniniai tinklai (*bidirectional recurrent neural networks*).

Aukštos kokybės balso aktyvavimo sistema turi tenkinti tokias savybes: aukštas reikšminių frazių aptikimas, priimtinas klaidingų aktyvavimų skaičius, galimybė veikti vartotojo įrenginyje, atsparumas triukšmui ir kalbos kintamumui, mažas uždelsimas tarp pagrindinės frazės ištarimo ir sistemos aktyvavimo.

Garsas yra nepertraukiamas fizinis mechaninių virpesių perdavimo fenomenas akustinės bangos pavidalu. Tačiau dauguma mašininio mokymosi metodų veikia su diskrečiais duomenimis. Šiuo atveju garso funkcijų išgavimas iš duomenų yra atliekamas bent dviem tikslais: garso pateikimas pavidalu, tinkamu taikyti mašininio mokymosi modelius, kuo didesnis informacijos, reikalingos problemai spręsti, išsaugojimas ir neįtraukimas informacijos, kuri nesusijusi su užduotimi.

Daugumoje balso aktyvavimo sistemų naudojami arba mel dažniniai cepstraliniai koeficientai (*mel-frequency cepstral coefficients* arba MFCC), arba filtrų banko energijos logoritmai (*logarithm of filter bank energy* arba LFBE).

Akustinio modelio uždavinys – pasirinkto akustinio vieneto garso savybių modeliavimas. Pavyzdžiui, akustinis modelis gali pateikti skaitinio MFCC vektoriaus tikimybių pasiskirstymą, su sąlyga, kad buvo ištarta konkreti fonema. Praktikoje akustinis modelis dažniausiai naudojamas apskaičiuoti $P(S|u)$, čia S – garsas, o u – tam tikras akustinis vienetas. Dažniausiai lengviau yra apskaičiuoti $P(u|S)$ ir pritaikius Bajeso teoremą gauti $P(S|u)$.

Populiariausi akustiniai modeliai yra Gauso mišinio modelis (*gaussian mixture model*), neuroniniai tinklai ir šių metodų derinys su paslėptais Markovo modeliais (*hidden Markov model*). Nenuostabu, kad akustinio modelio kokybė yra reikšminių žodžių aptikimo kokybės pagrindas, todėl dažnai didžioji balso aktyvavimo sistemos resursų dalis skiriama skaičiavimams naudojant akustinį modelį. Todėl šio modulio paspartinimui skiriama nemažai mokslinių darbų.

Svarbus klausimas, veikiantis balso aktyvavimo sistemos kokybę, yra tinkamo akustinio vieneto parinkimas. Sistemos kūrėjas turi rasti kompromisą tarp didelių vienetų (pavyzdžiui, žodžių), kuriuos lengviau rasti garso sraute dėl santykinio stabilumo, ir mažesnių vienetų (pavyzdžiui, fonemų), kurie yra lankstesni, nes leidžia pakartotinai naudoti informaciją

tarp skirtingų reikšminės frazės dalių. Kompromiso sprendimas gali būti ir per vidurį: kai kurie tyrinėtojai naudoja skiemenis ar žodžių dalis. Kitas variantas – naudoti dar mažesnius vienetus: fonemos dalis.

Dekodavimas naudojamas informacijai iš akustinio modelio kaupti ir nuspręsti, ar duotu momentu buvo ar nebuvo ištarta reikšminė frazė. Paprasčiausiu atveju tereikia palyginti tikimybę, apskaičiuotą naudojant akustinį modelį su pasirinktu slenksčiu, pavyzdžiui, kai akustinis vienetas yra visa reikšminė frazė. Priešingu atveju gali būti taikomi sudėtingesni algoritmai: Viterbi algoritmas arba *forward-backward algoritmas*.

Balso aktyvavimo sistemos veikimui įvertinti galima naudoti daugybę metrikų. Šias metrikas galima suskirstyti į tris grupes: aptikimo kokybės metrikos, veikimo greitis, sunaudotos RAM kiekis ir CPU.

Reikėtų pažymėti, kad skirtinguose tyrimuose aptikimo kokybei palyginti naudojamos skirtingos metrikos. Be to, metrikos dažnai yra nepalyginamos tarpusavyje dėl naudojamų skirtingų duomenų rinkinių ir reikšminių žodžių. Šie faktoriai apsunkina tinkamą gautų sistemų palyginimą.

Yra keletas krypčių, kaip kurti aktyvavimo balsu sistemas, kurios ne taip gerai dera prie siūlomos struktūros. Visų pirma, tai yra metodai, pagrįsti garso segmento palyginimu su reikšminės frazės tarimo pavyzdžiu. Beveik visada tokios balso aktyvavimo sistemos naudoja dinaminį laiko deformavimą (*dynamic time warping* arba DTW), kad apskaičiuotų atstumą tarp šablono ir bandomojo egzemplioriaus. Kitas būdas yra diskriminacinis reikšminių žodžių nustatymas (*discriminative keyword spotting*). Taikant šį metodą, garso segmentas yra įterpiamas į skaitmeninę požymių erdvę. Klasifikatorius yra apmokomas šioje erdvėje atskirti pavyzdžius su reikšminiu žodžiu ir be jo.

Siūloma tipinės balso aktyvavimo sistemos apžvalga ir struktūra buvo paskelbta tarptautiniame mokslo žurnale (Kolesau & Šešok, 2020c).

2. Balso aktyvavimo kokybės gerinimas keičiant sistemos dalis, nekeičiant sistemos neuroninio tinklo

Šiame skyriuje pateikiamas eksperimentinis kai kurių garso požymių išgavimo variantų tyrimas ir įvertinama šių variantų hiperparametrų įtaka galutinei balso aktyvavimo sistemos kokybei. Be to, buvo išanalizuota išgaunamų garso požymių supaprastinimo tendencija, tačiau parodyta, kad šiuo metu *a priori* žinios yra būtinos geresniems rezultatams pasiekti. Taip pat parodyta, kad koreguojant kadro ilgį (*frame length*) galima pasiekti iki 1 % didesnį tikslumą, palyginus su pagal numatytuosius parametrus nustatytomis, naudojamomis atvirojo kodo sistemose.

Kaip buvo aptarta ankstesniame skyriuje, dauguma mašininio mokymosi metodų veikia naudojant diskrečius duomenis, todėl garsui pateikti taikomi įvairūs būdai pavaizduoti garsą kaip laiko eilutę arba 2D vaizdą. Daugumoje darbų šiuo metu naudojamas MFCC, tačiau pastebima tendencija supaprastinti šį balso aktyvavimo sistemos modulį. Tai įmanoma naudojant galingesnius akustinius modelius. Kyla klausimas: ar jau atėjo laikas dar labiau supaprastinti garso požymius? Be to, galbūt tikslesnis hiperparametrų parinkimas, visų pirma, kadro ilgio, ir atstumas tarp gretimų kadrų pradžių (*frame stride*) leidžia pagerinti esamų sistemų kokybę. Dėl šių priežasčių šiame skyriuje aptariami šie metodai ir jų hiperparametrų įtaka balso aktyvavimo sistemos kokybei: MFCC, LFBE, spektrograma.

Kalbant apie skaičiavimo išteklius, MFCC yra brangiausias metodas, paskui eina LFBE. Spektrogramos apskaičiavimas resursų prasme yra pats paprasčiausias iš tiriamų metodų, tačiau jį taikant neatsižvelgiama į žmogaus kalbos veikimo ypatumus.

Eksperimentiniam garso požymių skaičiavimo metodų palyginimui buvo naudojamas *Google Speech Commands* duomenų rinkinys (Warden, 2018). Jį sudaro 105,829 audiofailai, 9,981 naudojamas validacijai, 11,005 – testavimui. Modelio užduotis yra klasifikuoti vienos sekundės garso failus į 10 klasių: „yes“, „no“, „up“, „down“, „left“, „right“, „on“, „off“, „stop“, „go“ arba pasakyti, kad garso įraše nėra nei vieno iš anksčiau paminėtų žodžių.

Buvo naudojamas konvoliucinis neuroninis tinklas *cnn-trad-fpool3* architektūros iš (Sainath & Parada, 2015). Modelis susideda iš dviejų konvoliucinių sluoksnių. Sujungimo sluoksnis (*pooling layer*) taikomas po kiekvienos konvoliucijos, siekiant išlyginti kalbos kintamumą laiko ir dažnio erdvėje. Ištiesintas linijinis vienetas (*rectified linear unit* arba ReLU) buvo naudojamas kaip aktyvavimo funkcija. Apmokymo metu buvo pritaikytas *Dropout* su koeficientu 0,5. Modelis turi 244,2 K ruošiamų parametrų. Neuroninis tinklas buvo optimizuotas naudojant stochastinį gradiento nusileidimą (*stochastic gradient descent*).

Visų pirma, buvo atliktas eksperimentas su pagal numatytuosius parametrus nustatytosiomis hiperparametrų reikšmėmis visiems garso požymių skaičiavimo metodams, išskyrus koeficientų kiekį (skaitinio vektoriaus ilgį) ir treniruočių partijos dydį. Iš atliktų eksperimentų buvo padarytos šios išvados:

- nepaisant to, kad spektrogramoje yra daugiau duomenų apie pradinį signalą (kadangi ir MFCC, ir LFBE gaunami naudojant informaciją prarandančias transformacijas iš garso spektrogramos), ją panaudojus pasiekiami prastesni rezultatai: dviejų sluoksnių konvoliucinis neuroninis tinklas negali išgauti reikiamo signalo iš garso požymių;
- MFCC beveik visada duoda geresnių rezultatų nei LFBE, tačiau skirtumas nėra labai didelis;
- bendruoju atveju yra prasminga naudoti kuo didesnį mokymo partijos dydį.

Po to buvo užfiksuotas MFCC veiksmų algoritmas ir maksimalus mokymo partijos dydis, nes toks požiūris davė geriausius rezultatus ankstesnėje eksperimentų serijoje. Naujoje serijoje mes ištyrėme tokius parametrus kaip kadro trukmė ir atstumas tarp gretimų kadrų pradžių:

- 10 ms atstumas labiau tinka balso aktyvavimo sistemoms nei 20 ms atstumas. Matyt, taip yra dėl to, kad esant didesniam atstumui sumažėja bendras kadrų skaičius, dėl to prarandama dalis informacijos apie originalų garsą (neskaičiuojant galimo kadrų dalinio sutapimo);
- ilgesnis kadro ilgis duoda geresnius rezultatus. Trūkumas yra galimas balso aktyvavimo atsakymo laiko sumažėjimas.

Tokiu būdu buvo parodyta, kad tolesnis garso funkcijų ištraukimo modulio supaprastinimas prieš apskaičiuojant spektrogramą šiuo metu yra nepagrįstas dabartiniams konvoliuciniams akustiniams modeliams. Tačiau kruopščiai parinkus MFCC ir LFBE hiperparametrus, galima pagerinti klasifikavimo kokybę. Optimalios kadro ilgio (*frame length*) reikšmės perrinkimas leidžia gauti iki 1 % santykinį aptikimo tikslumo pagerėjimą, naudojant *Google*

Speech Commands (Warden, 2018) duomenų rinkinį ir konvoliucinius neuroninius tinklus, palyginus su numatytosiomis 20 ms arba 30 ms reikšmėmis.

Toliau nagrinėjami akustinių vienetų pasirinkimo rezultatai balso aktyvavimo sistemai sukurti. Naudojant fiksuotą duomenų rinkinį ir akustinį modelį, šis pasirinkimas gali labai paveikti galutinę reikšminės frazės aptikimo kokybę. Norint suskaidyti pagrindinę frazę į paprastus akustinius vienetus, reikia iš anksto žinoti kalbos struktūrą. Tokios žinios gali palengvinti aptikimo uždavinį, t. y. gaunama kokybė gali pagerėti, tačiau nedidelės šio uždavinio klaidos gali nukreipti modelį klaidingu keliu. Buvo atliktas eksperimentinis tyrimas, kaip fonemų, skiemenų ar žodžių, akustinių vienetų pasirinkimas veikia balso aktyvavimo sistemos kokybę. Be to, buvo pasiūlyti du nauji sintetiniai akustiniai vienetai ir palyginti su anksčiau pateiktais.

Fonologijoje fonema yra garso vienetas, išskiriantis bent vieną žodžių porą tam tikroje kalboje. Pavyzdžiui, anglų kalboje žodžiai „sin“ ir „sing“ skiriasi garsu, kuris raštu žymimas raide „g“, todėl yra fonema „g“. Tarptautinė fonetinė abėcėlė (*International Phonetic Alphabet*) yra fonetinės abėcėlės, apimančios įvairių kalbų fonemas, pavyzdys.

Mokymo duomenų rinkinyje bet kuri tikslinės frazės fonema gali atsirasti ne tik kaip šios frazės dalis, bet ir kituose žodžiuose. Atsiranda keli variantai, kaip mokomajame rinkinyje panaudoti tokius fonemų pasireiškimus. Buvo apžvelgti keli požiūriai: tik „savų fonemų“ panaudojimas, t. y. tik tų pavyzdžių panaudojimas, kurie yra reikšminės frazės dalis, kaip akustinio modelio mokymo pavyzdžiai, „visų fonemų“ panaudojimas, t. y. net ir tų fonemų pavyzdžių naudojimas, kurie sudaro reikšminę frazę, kai jie pasitaiko kituose žodžiuose.

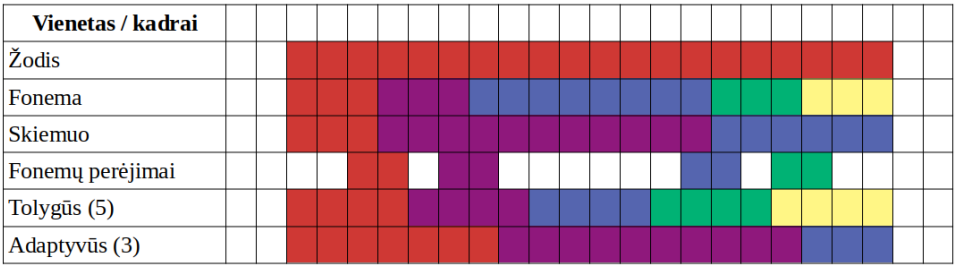
„Savų fonemų“ panaudojimas skatina kalbos modelį atsižvelgti į fonemos kontekstą, tai gali būti naudinga, jei mokymo rinkinys yra didelis ir (arba) modelis turi daug parametrų. Kita vertus, taikant tokį požiūrį, tikslinis akustinis vienetas jau nebėra fonema, o tai gali lemti mokymosi proceso pablogėjimą: juk, pavyzdžiui, fonema „a“ skirtinguose žodžiuose skamba labai panašiai. Tai reiškia, kad modelis bus priverstas išmokti ne tik šios fonemos tarimą, bet ir apytikslius visos reikšminės frazės požymius. Naudodamas „visas fonemas“ modelis gauna daugiau tikslinių akustinių vienetų pavyzdžių, bet neturi paskatų naudoti kontekstinę informaciją, kuri gali būti naudinga.

Skiemuo yra kalbos garsų sekos organizavimo vienetas. Tipiškai skiemenį sudaro skiemens branduolys – dažniausiai balsė ir nebūtinos pradinės ir galutinės paraštės (*margins*) – dažniausiai priebalsiai. Žodžio skiemenų skaičius beveik visada yra mažesnis už fonemų skaičių, todėl dekoduoti yra lengviau, tačiau skiemens ilgis yra labiau kintamas, tai apsunkina akustinio modelio uždavinį. Kitas apžvelgtas variantas yra naudoti visą žodį kaip akustinį vienetą. Šiuo atveju akustinis modelis turėtų turėti tik du išėjimus: tikimybę, kad šiuo metu ištariamas tikslinis žodis, ir priešingo įvykio tikimybę. Be to, perėjimai tarp fonemų gali būti naudojami kaip akustiniai vienetai, nes būtent perėjimų momentais akustinis vaizdas kardinaliai pasikeičia, o tai gali būti naudingesnis modeliavimo tikslas.

Žodžių skaidymas į fonemas atliekamas su tam tikromis prielaidomis apie kalbos sandarą. Vietoj to galima naudoti nuo kalbos nepriklausomą skaidymą ir leisti neuroniniam tinklui „išmokti“ reikalingą informaciją iš duomenų. Todėl siūlomi du nauji akustiniai vienetai. Pirmas iš jų – „tolygus“ (*uniform unit*). Šio akustinio vieneto ribose daroma prielaida, kad reikšminį žodį sudaro n vienodos trukmės akustinių vienetų, o n yra hiperparametras.

Fonemos nebūtinai būna vienodos trukmės. Todėl „tolygių“ akustinių vienetų prigimtis gali būti labai skirtinga: kelios fonemos, viena fonema, fonemos dalis arba perėjimas tarp

fonemų. Atsižvelgiant į tai, buvo pasiūlytas „adaptyvus“ akustinis vienetas. Šiuo atveju žodį padaliname į n dalių, sumažindami atstumų sumas nuo kiekvieno kadro iki vidutinio kadro šiame vienete LFBE požymių erdvėje. Tokį skaidymą galima efektyviai apskaičiuoti naudojant dinaminį programavimą per $\mathcal{O}(m^2 n)$, čia m yra žodžio ilgis kadruose. Kaip ir „tolygių“ vienetų atveju, n yra hiperparametras. Visų apžvelgtų vienetų iliustraciją pagal kadrus galima pamatyti S2.1 paveikslėlyje.



S2.1 pav. Akustinių vienetų pasiskirstymo per kadrus pavyzdys

Eksperimentams buvo naudojamas duomenų rinkinys iš 174 639 garso įrašų rusų kalba. Buvo panaudoti šie reikšminiai žodžiai: „Алиса“ (5 fonemos, 3 skiemenys, 7455 bandomieji pavyzdžiai), „включи“ (6, 2, 542), „тебя“ (4, 2, 462), „мне“ (3, 1, 889).

Eksperimentų rezultatai pateikti S2.1 lentelėje. Modelis buvo apmokytas ir aktyvavimo slenkstis pasirinktas tokiu būdu, kad klaidingo atmetimo rodiklis (*false reject rate*) būtų 0.1, o klaidingų aktyvavimų dalis (*false alarm rate*) būtų kuo mažesnė. Skliausteliuose nurodoma hiperparametro n reikšmė (tiems pavyzdžiams, kur tai yra prasminga).

S2.1 lentelė. Klaidingų aktyvavimų proporcijos (*false alarm rate*) pablogėjimas, palyginti su geriausiu nurodyto žodžio variantu

Vienetas / žodis	„Алиса“	„включи“	„тебя“	„мне“
savo / visos fonemos	5 % / 36 %	1 % / **0 %**	N/A / **0 %**	54 % / **0 %**
savo / visi skiemenys	17 % / 34 %	150 % / 131 %	114 % / 105 %	141 % / 137 %
savo / visi perėjimai	49 % / 22 %	24 % / 53 %	123 % / 91 %	88 % / 102 %
tolygūs vienetai	**0 % (6)**	19 % (6)	95 % (3)	35 % (7)
adaptyvūs vienetai	10 % (5)	28 % (8)	152 % (2)	54 % (7)
žodis	80 %	733 %	176 %	139 %

Galiausiai buvo išnagrinėtas pasirinkimas, kai visi akustinio vieneto atvejai mokymo rinkinyje naudojami kaip mokymo tikslai (parinktis „visi“), ir variantas, kai naudojami tik tie akustinio vieneto atvejai, kurie yra žodžio dalis (parinktis „savo“). Galima pastebėti (žr. S2.2 lentelę), kad parinktis „savas“ yra tinkamesnė, kai duomenų rinkinys yra didelis arba tikslinis žodis yra gana ilgas. Tokiais atvejais modelis gali pasinaudoti kontekstine informacija. Kai duomenų rinkinys yra mažas, pasirenkant „savo“ parinktis labai sumažėja kiekvienos klasės teigiamų pavyzdžių skaičius, todėl optimizuoti tampa sunku. Dėl tos pačios priežasties variantas „žodis“ nerodo gerų rezultatų, nes dauguma pavyzdžių yra neigiami. Bendruoju atveju pirmenybė tūri būti teikiama parinkčiai „visi“.

S2.2 lentelė. Klaidingų aktyvavimų proporcijos (*false alarm rate*) pablogėjimas, palyginti su geriausiu nurodyto žodžio variantu

Vienetas / žodis	„Алиса“	„включи“	„тебя“	„мне“	Bendras
savo	23 %	58 %	118 %	94 %	73 %
visos	31 %	61 %	65 %	80 %	60 %

Iš atliktų eksperimentų galima padaryti tokias išvadas: fonemos yra geras bazinis pasirinkimas, ypač jei mokymo pavyzdžių yra nedaug; siūlomi „uniforminiai“ vienetai vidutiniškai užima antrą vietą po „fonemų“, tačiau rodo geriausius rezultatus esant daugybei mokymo pavyzdžių; tikslumas yra vidutiniškai 13 % didesnis, jei naudojami visi akustinių vienetų mokymo pavyzdžiai, palyginus su variantu, kai naudojami tik tie pavyzdžiai, kurie yra tikslinių reikšminių žodžių dalis.

Toliau siūlomas metodas, kaip pagerinti klaidingų atmetimų dažnį (*false reject rate*) reikšminiam balso aktyvavimo sistemos naudojimui. Tokių sistemų vartotojai dažnai kartoja aktyvavimo žodį, jei jis buvo klaidingai praleistas. Siūloma dekodavimo modulio modifikacija, tinkanti kai kurioms fonemomis pagrįstoms balso aktyvinimo sistemoms. Atliktuose eksperimentuose tokia modifikacija leido aptikti 14,6 % anksčiau praleistų pakartojimų, nepadidinant klaidingų atmetimų dažnio. Leidus 1 % naujų klaidingų aktyvavimų, galima rasti jau 79 % anksčiau praleistų pakartojimų. Buvo panaudotas faktas, kad reikšminio žodžio pakartojimas (kartu su originaliu tarimu) turi dvigubai daugiau fonemų nei pavienis tarimas, o tai leidžia patikimiau aptikti tokius atvejus.

Balso aktyvinimo sistemos turi dviejų tipų aptikimo klaidas: klaidingi aktyvavimai, klaidingi praleidimai (aptikimo praleidimas, kai žodis buvo ištartas). Todėl aptikimo kokybė dažnai matuojama naudojant porą metrikų: klaidingo atmetimo dalis (FRR) ir klaidingo aktyvavimo dalis (FAR). Nagrinėjamos modifikacijos tikslas – pagerinti FRR. Vienas iš siūlomo metodo privalumų yra tai, kad akustinis modelis nėra permokomas, o tai sumažina reikiamą pavyzdžių skaičių hiperparametrų parinkimui ir nereikalauja didelių skaičiavimo resursų. Eksperimentams atlikti buvo naudojamas privatus rusų kalbos duomenų rinkinys, sudarytas iš 90 000 garso failų, kurių vidutinė trukmė yra 6 sekundės. Apie 40 % pavyzdžių turėjo bent vieną reikšminio žodžio tarimą. Duomenų rinkinys buvo padalintas į dvi lygias dalis: hiperparametrams parinkti ir galutinėms metrikoms matuoti.

Siekiant įvertinti siūlomą modifikaciją, buvo naudojama balso aktyvavimo sistema, kuri naudoja neuroninį tinklą kaip akustinį modelį, LFBE požymius ir dekodavimo modulį, siūlomą (Chen et al., 2014a). Šis modulis pirmiausia išlygina triukšmingus neuroninio tinklo išėjimus, apskaičiuodamas tikimybių vidurkį per slenkantį langą (*sliding window*), kurį galima padaryti per $\mathcal{O}(n_i n_f)$, čia n_i yra neuroninio tinklo išėjimų skaičius, o n_f yra kadrų skaičius. Paskui atliekamas reikšminio žodžio ištarimo tikimybės skaičiavimas naudojant išlygintas tikimybes: žodžio ištarimo tikimybė laikoma didžiausiu geometriniu akustinių vienetų, sudarančių žodį, vidurkiu, pagal visus galimus būdus reikia parinkti akustinių vienetų padėtis kadrų segmente. Naudojant dinaminį programavimą, visas n_f kadrų dekodavimo modulis gali būti realizuotas per $\mathcal{O}\left(n_f(n_i + w_{\max} n)\right)$, čia n_i yra tinklo išėjimų skaičius, $w_{\max}$ yra segmento ilgis kadrais, n yra akustinių vienetų skaičius žodyje.

Siūlomo metodo esmė yra, be pradinės akustinių vienetų sekos, ieškoti ir dvigubos sekos, taip imituojant reikšminio žodžio kartojimo situaciją. Žinoma, šios tikimybės skaičiavimas užima papildomą laiką, tačiau praktiškai dominuoja laikas, kurio reikia akustiniam

modeliui apskaičiuoti. Siūloma naudoti ir viengubo, ir dvigubo tarimo tikimybes, kiekvienai situacijai naudojant skirtingas slenksčių reikšmes. Atkreipkime dėmesį, kad norint apskaičiuoti dvigubo tarimo tikimybę, reikia padidinti $w_{\max}$ reikšmę, nes natūralu, kad toks tarimas trunka ilgiau nei viengubas. Maža to, prasminga ją padidinti daugiau nei dviem kartais, nes tarp pakartojimų paprastai būna pauzė. Nesant naujų (palyginus su originalia sistema) klaidingų aktyvavimų, modifikacija leidžia rasti 14,6 % praleistų pakartojimų ir 79 % su 1 % naujų klaidingų aktyvavimų.

Siekiant įvertinti resursų sąnaudas taikant siūlomą metodą, buvo paleista originali sistema ir sistema su modifikacija 1 valandai garso viename Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz branduolyje. Galiausiai sunaudotas laikas išaugo 3 % ir sudarė 79 sekundes.

Šio skyriaus rezultatai buvo pristatyti tarptautinėje konferencijoje ir publikuoti šios konferencijos medžiagoje (Kolesau & Šešok, 2020a, 2021, 2021a).

3. Balso aktyvavimo sistemos apmokymas mažų išteklių sąlygomis

Balso aktyvavimo sistemos sprendžia iš anksto nustatyto reikšminio žodžio ar frazės radimo garso sraute problemą. Kadangi sunku suformuluoti algoritmą, leidžiantį nustatyti, ar pagrindinė frazė buvo ištarta, ar ne, nenuostabu, kad šiai problemai spręsti jau seniai taikomi euristiniai metodai ir mašininio mokymosi metodai.

Viena iš pagrindinių šiuolaikinių sprendimų problemų yra būtinybė naudoti labai didelius duomenų rinkinius neuroniniam tinklui treniruoti. Pavyzdžiui, Chen et al. (2014a) kiekvienam reikšminiam žodžiui naudoja šimtus tūkstančių pavyzdžių, o Jose et al. (2020) naudoja milijonus. Kita vertus, nėra akivaizdaus būdo sukurti aukštos kokybės balso aktyvavimo sistemą, kai yra mažai mokymo duomenų, pavyzdžiui, tokioms kalboms kaip lietuvių. Ši situacija kelia klausimą, kaip tokiu atveju konstruoti balso aktyvavimo sistemas. Tai gali būti naudinga dėl kelių priežasčių:

- gaminio pritaikymas naudojant vartotojo nurodytus reikšminius žodžius ar frazes;
- sistemų kūrimas kalboms, kuriose yra mažai pasiekiamų duomenų.

Šiame skyriuje nagrinėjami keli gerai žinomi ir siūlomi nauji būdai, kaip pagerinti balso aktyvavimo sistemos kokybę mažame pažymėtame duomenų rinkinyje, palyginus su standartiniu prižiūrimo mokymosi (*supervised learning*) metodu. Šiame skyriuje duomenų rinkinys vadinamas mažu, jei kiekvienam reikšminiam žodžiui apmokymo dalyje naudojama ne daugiau kaip 20 pavyzdžių.

Uždaviniams spręsti buvo parengtas ir naudojamas nedidelis duomenų rinkinys lietuvių kalbai. Visuose eksperimentuose buvo naudojama ta pati neuroninių tinklų architektūra, tačiau buvo pridėti keli garso požymių ištraukimo modulio pakeitimai, taikomi tam tikri metodai akustiniam modeliui paruošti arba keičiamas neuroninio tinklo mokymo būdas.

Eksperimentams buvo naudojami trys duomenų rinkiniai:

- rinkinys anglų kalbai *Google Speech Commands dataset (GSC)* (Warden, 2018);
- privatus rinkinys rusų kalbai;
- originalus surinktas ir parengtas rinkinys lietuvių kalbai.

Google Speech Commands dataset buvo išleistas 2017 m. pagal licenciją *Creative Commons*. Rinkinyje yra apie 100 000 vienos sekundės trukmės įrašų, kurių kiekviename yra vienas iš 30 žodžių, pasakytų tūkstančių skirtingų žmonių, taip pat foniniai triukšmai, tokie kaip rožinis triukšmas, baltas triukšmas, kalbos triukšmas. Kaip ir *Google* darbuose, mes sprendėme klasifikavimo į 12 klasių uždavinį: „yes“, „no“, „up“, „down“, „left“, „right“, „on“, „off“, „stop“, „go“, nežinomas žodis ir tyla.

Rusų kalbos duomenų rinkinį sudaro 400 000 vienos sekundės įrašų, kuriuose ištariamas vienas iš 80 žodžių. Įraše, kuris buvo daromas į mobiliuosius telefonus, dalyvavo apie 100 žmonių. Šiame rinkinyje nėra triukšmų, todėl pakartotinai panaudojome juos iš GSC. Sprendžiame klasifikavimo į 12 klasių uždavinį: „один“ (vienas), „два“ (du), „три“ (trys), „четыре“ (keturi), „пять“ (penki), „да“ (taip), „нет“ (ne), „спасибо“ (ačiū), „стоп“ (stop), „включи“ (įjunk), nežinomas žodis ir tyla.

Galiausiai, specialiai savo eksperimentams ir tolesniems balso aktyvavimo sistemų tyrimams mažų duomenų rinkinių atveju buvo surinktas nedidelis duomenų rinkinys lietuvių kalbai. Šį rinkinį sudaro 28 savanorių užrašai, iš kurių kiekvienas padiktavo po 20 žodžių į mobilųjį telefoną. Šie įrašai buvo suskirstyti į vienos sekundės atkarpas. Segmentai tarp žodžių buvo naudojami kaip foniniai triukšmai. Juose yra tyla, kalbos artefaktai ir kiti triukšmai, pavyzdžiui, pravažiuojančio automobilio triukšmas. Buvo sprendžiama klasifikavimo į 15 klasių problema: „ne“, „ačiū“, „stop“, „įjunk“, „išjunk“, „į viršų“, „į apačią“, „į dešinę“, „į kairę“, „startas“, „pauzė“, „labas“, „iki“, nežinomas žodis ir tyla.

Buvo naudojami dviejų tipų garso požymiai: standartiniai LFBE požymiai arba mašininiu būdu apmokyti *wav2vec* požymiai, pasiūlyti (Schneider et al., 2019) kalbos atpažinimo problemai spręsti. Siūlomas būdas, kaip juos panaudoti, kad išspręstume balso aktyvavimo problemą, kai yra nedidelis išteklių kiekis.

Siekiant įvertinti, ar siūlomi metodai tinka skirtingoms neuroninių tinklų architektūroms, visi eksperimentai buvo atlikti su trijų sluoksnių visiškai sujungtu tinklu ir kelių tipų likutiniais neuroniniais tinklais (*residual neural networks*). Norint gauti patikimus rezultatus, kiekvienam eksperimentui buvo atlikta atsitiktinė hiperparametrų paieška.

Bazinis variantas: standartinis neuroninio tinklo mokymas tiksliniame duomenų rinkinyje, atliekamas LFBE garso požymių pagrindu.

wav2vec: garso požymiams išgauti buvo naudojamas iš anksto parengtas wav2vec modelis. Siūlomas metodas pagerino klasifikavimo kokybę, jei duomenų rinkinyje kiekvienam reikšminiam žodžiui buvo naudojamas tik nedidelis pavyzdžių skaičius. Naudojant (didelį) išsamų, visą duomenų rinkinį, pagerėjimas išnyksta. Įdomus faktas yra tai, kad pagerėjimas pastebimas visoms kalboms, nepaisant to, kad *wav2vec* modelis buvo apmokytas tik anglų kalbai.

Tikslus (*fine-tuning*) iš anksto paruošto modelio derinimas prie tikslinio duomenų rinkinio: šis populiarus metodas iš pradžių siūlo apmokyti modelį dideliame duomenų rinkinyje, o vėliau papildomai apmokyti modelį (mažam) tiksliniam duomenų rinkiniui. Pagrindinė idėja yra ta, kad klasifikatorius išmoks neakivaizdžius didelio duomenų rinkinio požymius ir galės juos panaudoti sprendžiant tikslinę problemą. Išankstiniam apmokymui mes naudojome *Google Speech Commands dataset* duomenų rinkinį, o derinimui – duomenų rinkinį lietuvių kalbai.

Egzempliorinis (*exemplar-like*) išankstinis mokymas: šis metodas yra populiarus sprendžiant kompiuterinio regėjimo problemas. Buvo pasiūlyta jį naudoti balso aktyvavimo uždaviniui. Pagrindinė idėja yra pasirinkti n atsitiktinių „pradinių“ pavyzdžių, sugene-

ruoti k dublikatus kiekvienam iš šių pavyzdžių, įterpiant triukšmus arba augmentacijas. Gautame duomenų rinkinyje apmokomas klasifikatorius, kuris turi atskirti, iš kurio „pradinio“ pavyzdžio buvo gautas įvesties pavyzdys. Vėliau šis klasifikatorius suderinamas (*fine-tune*) sprendžiamam uždaviniui tiksliniame duomenų rinkinyje. Pagrindinė metodo motyvacija yra tai, kad klasifikatorius išankstinio mokymo metu yra priverstas išmokti atskirti būdingus srities, kurioje jis dirba, požymius. Metodo naudingumas priklauso nuo „pradinių“ pavyzdžių pasirinkimo kokybės ir sėkmingo augmentacijų pasirinkimo.

Savarankiškas mokymasis (*self-training*): šio metodo idėja yra apmokyti modelį mažame duomenų rinkinyje, o vėliau pritaikyti šį modelį žymėti didelį duomenų rinkinį be žymų. Gautame dideliame duomenų rinkinyje (su, galbūt, neteisingomis žymėmis) treniruojamas kitas modelis ir procesas kartojamas iki konvergavimo. Šis metodas gerai tinka kalbos atpažinimo sistemoms.

Bendras mokymasis su keliais duomenų rinkiniais: taikant šį siūlomą metodą, vienoje treniruotėje naudojamas didelis pažymėtas duomenų rinkinys (*Google Speech Commands dataset*) kartu su mažu tiksliniu duomenų rinkiniu. Modelis išmoksta atskirti reikšminius žodžius kombinuotame rinkinyje. Buvo pastebėta, kad mokymosi procese šiuo atveju reikia atsižvelgti į tai, kad pradiniai duomenų rinkiniai yra skirtingų dydžių, kitaip modelis koncentruojasi ties didesniu rinkiniu.

Eksperimentų rezultatai apibendrinti S3.1 lentelėje.

S3.1 lentelė. Tirtų metodų klasifikavimo tikslumas. Rezultatai, geresni už bazinį LFBE sprendinį, pateikti paryškintu šriftu

Metodas / arhitektūra	ff	res8	res15	res26
Bazinis sprendinys	72,31 %	78,46 %	89,23 %	80,00 %
wav2vec	**78,46 %**	**90,77 %**	86,15 %	**83,08 %**
Tikslus derinimas	N/A	**89,23 %**	**90,77 %**	N/A
Egzempliorinis išankstinis mokymas	N/A	**80,00 %**	**90,77 %**	N/A
Savarankiškas mokymasis	63,08 %	**81,54 %**	86,15 %	**81,54 %**
Bendras mokymasis	**84,62 %**	**84,62 %**	**93,85 %**	**90,77 %**

Atlikto tyrimo tikslas buvo pagerinti rezultatus nedideliam duomenų rinkiniui lietuvių kalbai, palyginus su baziniu sprendiniu šiam duomenų rinkiniui. Iš anksto paruoštų *wav2vec* garso požymių naudojimas – vienas iš pasiūlytų sprendimų – leidžia pasiekti užsibrėžtą tikslą. Kitas būdas yra taikyti populiarų metodą reguliuojant modelį, kuris buvo iš anksto paruoštas naudojant kitą duomenų rinkinį.

Egzempliarinis išankstinis mokymasis parodė rezultatus, panašius į bazinį sprendinį, tačiau nedavė reikšmingo pagerėjimo. Buvo padaryta prielaida, kad taip atsitiko todėl, kad pasirinktos augmentacijos neleido modeliui išmokti požymius, kurie nepriklauso nuo kalbančiojo pakeitimo.

Savarankiškas mokymasis parodė santykinį pagerėjimą, tačiau *wav2vec* naudojimas beveik visada duoda geresnius rezultatus. Tikėtina, kad taip atsitiko dėl nesėkmingo nepažymėtų duomenų parinkimo, siekiant interaktyviai pagerinti modelių kokybę: jame beveik nebuvo reikšminių žodžių tarimų.

Galiausiai, naudojant didelius pažymėtus duomenų rinkinius bendram mokymuisi (o ne tik išankstiniam mokymuisi), pasiekiamas didžiausias pagerėjimas, bet tik tuo atveju, kai tinkamai parinkti hiperparametrai.

Atlikus šiuos tyrimus buvo surinktas duomenų rinkinys lietuvių kalbai tolimesniam tyrimui ir tobulinimui bei išanalizuoti keli (taip pat ir naujai pasiūlyti) balso aktyvavimo sistemų konstravimo metodai kalboms, kurios turi mažai paruoštų duomenų. Buvo pasiektas 10 % aktyvavimo tikslumo pagerėjimas, kai kiekvienam reikšminiam žodžiui buvo panaudoti ne daugiau kaip 7 pavyzdžiai, ir 29 %, jei ne daugiau nei 5, naudojant iš anksto paruoštus *wav2vec* garso požymius. Bendras mokymasis su duomenų rinkiniu, skirtu kitai kalbai, rodo aktyvavimo tikslumo pagerėjimą nuo 7 % iki 25 %, priklausomai nuo architektūros, ir rodo geriausią rezultatą tarp visų architektūrų su 93,85 % klasifikavimo tikslumu.

Šio skyriaus rezultatai buvo publikuoti tarptautiniuose žurnaluose (Kolesau & Šešok, 2020b, 2021b).

Bendrosios išvados

1. Atlikus literatūros analizę, kuri parodė, kad kokybiškos balso aktyvavimo sistemos kūrimo tema yra aktuali tiek mokslininkams, tiek pramonei. Įprasta šiuolaikinė balso aktyvavimo sistemos struktūra susideda iš: garso požymių ištraukimo modulio (dažniausiai tai yra MFCC arba LFBE), akustinio modelio (beveik visada tai yra neuroninis tinklas) ir dekodavimo modulio. Dauguma šiuolaikinių tyrimų yra orientuoti į neuroninio tinklo, kaip balso aktyvavimo sistemos akustinio modelio, tobulinimą.

2. Išankstinių žinių panaudojimas garso požymių ištraukimo modulyje šiuolaikinėse balso aktyvavimo sistemose yra pagrįstas: tai buvo parodyta lyginant 3 populiariausius garso požymių ištraukimo modulius: MFCC, LFBE ir garso spektrogramą. Naudojant MFCC, aptikimo kokybė yra 0,4 % geresnė negu naudojant LFBE, priklausomai nuo treniruočių partijos dydžio (angl. *batch size*) ir koeficientų kiekio. Bet globalus tikslumo maksimumas yra vienodas ir lygus 80,7 %. Globalus tikslumo maksimumas spektrogramai yra 67,5 %. Kadro ilgio reikšmės 40 ms ir 55 ms mel dažnio cepstralių koeficientų (*mel frequency cepstral coefficients*) metodui, *Google Speech Commands* duomenų rinkiniui (Warden, 2018) ir konvoliuciniams neuroniniams tinklams leidžia 0,95 % ± 0,77 % (95 % pasikliautinasis intervalas) pagerinti aptikimo kokybę, lyginant su numatytomis 20 ms ir 30 ms vertėmis.

3. Atlikti eksperimentai parodė, kad fonemų, kaip akustinio vieneto naudojimas, yra geriausias pasirinkimas pagal numatytuosius parametrus, ypač mažų duomenų rinkinių sąlygomis. Siūlomas „tolygus“ variantas vidutiniu atveju yra antras (atsilikimas sudaro apie 30 %) ir rodo geresnius rezultatus esant dideliam mokymo pavyzdžių skaičiui, tą galima paaiškinti indukciniu šališkumu. Taip pat buvo parodyta, kad aptikimo tikslumas yra 11,78 % ± 7,77 % (95 % pasikliautinasis intervalas) didesnis, kai visi treniruočių rinkinyje esantys akustinių vienetų pavyzdžiai naudojami kaip mokymo tikslai, palyginus su tuo atveju, kai naudojami tik tie akustinių vienetų pavyzdžiai, kurie yra reikšminio žodžio dalis.

4. Buvo pasiūlytas klaidingų praleidimų dažnio (FRR) pagerinimo metodas. Pasiūlytas balso aktyvavimo sistemų, pagrįstų foneminiais akustiniais vienetais, dekodavimo modulio modifikavimas leidžia aptikti 13,32 % ± 1,76 % (95 % pasikliautinasis intervalas) anksčiau praleistų reikšminio žodžio pakartojimų be naujų klaidingų

aktyvavimų. Leidus tik 1 % naujų klaidingų aktyvavimų, galima rasti 79,9 % ± 1,5 % (95 % pasikliautinasis intervalas) anksčiau praleistų pakartojimų. Siūlomas metodas nereikalauja pakartotinio neuroninio tinklo apmokymo ir padidina skaičiavimų laiką tokio pat ilgio garso įrašui blogiausiu atveju 4 %.

5. Pasiūlytas išankstinis garso požymių išgavimo modulio apmokymas naudojant neprižiūrimą išankstinį mokymą (*unsupervised pre-training*) pagerina balso aktyvavimo tikslumą (*accuracy*) 8,04 % ± 4,73 % (95 % pasikliautinasis intervalas), kai kiekvieno žodžio mokymo pavyzdžių skaičius neviršija 7, ir 24,68 % ± 6,24 % (95 % pasikliautinasis intervalas), kai yra ne daugiau kaip 5 mokymo pavyzdžiai lietuvių, anglų ir rusų kalboms. Tačiau pagerėjimas išnyksta naudojant didelius duomenų rinkinius, o tai gali reikšti siūlomo metodo apribojimus. Būdai, kaip pagerinti balso aktyvavimo sistemos kokybę naudojant mažus duomenų rinkinius, buvo adaptuoti iš kitų sričių. *Exemplar-like pre-training* yra metodas, kuris taikomas ir vaizdams apdoroti ir rodo rezultatus, panašius į gautus taikant bazinį metodą (tikslumas res15 architektūrai yra 90,77 %, palyginus su 89,23 % bazinio metodo, tačiau 75,38 % res8 architektūrai, palyginus su 78,46 % bazinio metodo). Buvo surinktas lietuviškas duomenų rinkinys, skirtas būsimiems eksperimentams su balso aktyvavimo sistemomis mažų duomenų rinkinių sąlygomis. Buvo pasiūlyti ir įvertinti balso aktyvavimo sistemų konstravimo metodai, skirti darbui su nedideliais duomenų rinkiniais. Geriausias iš pasiūlytų metodų palyginus su standartiniu baziniu sprendiniu parodė santykinį pagerėjimą 10,17 % ± 2,11 % (95 % pasikliautinasis intervalas) tiesioginio pasiskirstymo ir liekamųjų neuroninių tinklų architektūroms.

Annexes

Annex A. The Result Tables of Deep Learning Experiments

Table A1. Test metrics and hyperparameters on the Lithuanian dataset

Features	BS	S	Architecture	L	L'	Accuracy	CE loss
LFBE	32	512	ff	0.0017	5.1070	72.31%	1.241
wav2vec	16	256	ff	0.0021	1.4282	78.46%	0.746
LFBE	32	512	res15	0.1049	4.2465	89.23%	0.448
wav2vec	16	512	res15	0.0092	3.4831	86.15%	0.317
LFBE	64	512	res15-narrow	0.0744	5.2013	83.08%	0.504
wav2vec	64	2048	res15-narrow	0.0049	3.2217	80.00%	0.629
LFBE	32	1024	res26	0.0657	2.0357	80.00%	0.680
wav2vec	16	2048	res26	0.0055	6.8291	83.08%	0.408
LFBE	32	1024	res26-narrow	0.0695	6.2596	83.08%	0.492
wav2vec	16	256	res26-narrow	0.0299	3.0330	72.31%	0.726
LFBE	16	2048	res8	0.0114	2.9399	78.46%	0.610
wav2vec	32	512	res8	0.1233	9.8922	90.77%	0.130
LFBE	16	2048	res8-narrow	0.1734	8.3757	73.85%	0.752
wav2vec	64	2048	res8-narrow	0.0812	7.8581	84.62%	0.504

Table A2. Test metrics and hyperparameters on Google Speech Commands dataset (Warden, 2018)

Limit	Features	BS	S	Architecture	L	L'	Accuracy
no	LFBE	64	1024	ff	0.0018	4.0535	73.14%
no	wav2vec	32	1024	res26-narrow	0.0352	7.6938	91.45%
no	LFBE	64	1024	res8-narrow	0.0161	7.3268	92.61%
no	LFBE	16	2048	res8	0.0067	2.1576	95.07%
no	wav2vec	16	2048	res15-narrow	0.0198	6.6472	95.70%
no	wav2vec	32	2048	res8	0.0036	9.6358	95.79%
no	wav2vec	16	2048	res8-narrow	0.0164	1.2408	95.91%
no	wav2vec	64	512	res26	0.0042	5.3919	95.97%
no	wav2vec	64	1024	res15	0.0983	6.4306	96.07%
no	LFBE	32	2048	res15-narrow	0.0128	7.9980	96.44%
no	LFBE	32	2048	res26-narrow	0.1515	5.7268	96.75%
no	LFBE	64	512	res15	0.0035	1.4772	97.03%
no	wav2vec	16	2048	ff	0.0056	2.8667	97.20%
no	LFBE	64	2048	res26	0.0578	4.2733	97.46%
3	wav2vec	16	512	res8	0.0015	6.8410	37.05%
3	LFBE	64	32	res15	0.0419	5.5069	39.30%
5	LFBE	16	256	res15	0.2014	7.6873	45.91%
7	LFBE	16	256	res15	0.3133	4.3166	50.55%
10	LFBE	64	128	res15	0.0777	6.4338	54.40%
5	wav2vec	16	1024	res15-narrow	0.0050	7.6948	59.41%
7	wav2vec	64	512	res8-narrow	0.0300	4.5449	60.41%
10	wav2vec	16	2048	res8	0.0195	9.1938	61.05%
20	LFBE	16	2048	res15	0.0237	3.3050	74.22%
20	wav2vec	32	1024	res8-narrow	0.0099	4.4134	75.63%

Table A3. Test metrics and hyperparameters on the Russian dataset

Limit	Features	BS	S	Architecture	L	L'	Accuracy
3	LFBE	16	32	res15-narrow	0.1635	4.1339	31.72%
5	LFBE	16	256	res8-narrow	0.0312	4.9367	40.42%
3	wav2vec	16	256	res15	0.0179	1.8520	48.51%
10	LFBE	32	16	ff	0.1637	5.8915	54.60%
5	wav2vec	16	128	ff	0.0067	6.6971	57.65%
7	LFBE	16	512	res26	0.0060	7.8152	57.69%
10	wav2vec	32	512	res26	0.3383	2.8521	58.68%
7	wav2vec	16	1024	res15-narrow	0.0046	2.5055	63.88%
20	LFBE	16	2048	res15	0.0012	1.3961	67.43%
20	wav2vec	64	2048	res26	0.0122	4.5916	73.79%

Annex B. Research Information Sheet For Participants

Dalyvio vardas, pavardė, amžius:
Tyrimo pavadinimas: Balso aktyvavimo sistemų efektyvumo gerinimas naudojant mašininio mokymosi metodus

INFORMACIJOS APIE MOKSLINĮ TYRIMĄ LAPAS DALYVIUI

1. Kodėl atliekamas šis tyrimas?

Mokslinis tyrimas atliekamas mokslo publikacijų ir daktaro disertacijos rengimui.

2. Ar aš privalau dalyvauti tyrime?

Ne. Prieš priimdamas(-a) sprendimą dalyvauti ar ne, galite užduoti klausimus apie tyrimą

3. Kokia bus tyrimo eiga, jei aš sutiksiu dalyvauti tyrime?

Turėdami Jūsų sutikimą, norėtume padaryti garso įrašą. Garso įrašas bus reikalingas, kad būtų sukurtas garsynas lietuvių kalba.

4. Ar yra kokia nors rizika dalyvauti tyrime?

Norėdami sumažinti bet kokią galimą riziką, duomenis anonimizuosime.

5. Ar yra kokia nors nauda dalyvaujant tyrime?

Dalyvaudamas(-a) šiame tyrime Jūs negausite nei tiesioginės, nei asmeninės naudos.

6. Kaip bus valdomi surinkti duomenys?

Informacija, kurią pateikiate tyrimo metu, yra tyrimo duomenys. Bet kokie tyrimo duomenys, iš kurių galite būti identifikuoti – vardas, pavardė, lytis, amžius, garso įrašas yra traktuojami kaip asmens duomenys.

Tyrimui renkami duomenys patenka į specialiųjų kategorijų asmens duomenų kategoriją, tokią kaip biometriniai duomenys.

Neskelbtini duomenys bus saugomi 3 metus užtikrinus visas būtinas organizacines ir technines saugumo priemones.

Kiti tyrimų duomenys (įskaitant sutikimo formas) bus saugomi 3 metus po tyrimo rezultatų paskelbimo.

Tyrimo duomenys bus atverti https://github.com/kolesov93/lt_speech_commands ir bus prieinami visiems.

Tiriamasis turi teisę atšaukti sutikimą dėl asmens duomenų tvarkymo iki 2020 m. spalio 1 d.

Tyrėjas ir jo vadovas turės prieigą prie tyrimo duomenų. Atsakingiems Vilniaus Gedimino technikos universiteto nariams gali būti suteikta prieiga prie duomenų, skirtų tyrimams stebėti arba auditui atlikti, ir Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos ir / ar procedūrų pažeidimą.

7. Ar tyrimas bus paskelbtas?

Tyrimas gali būti paskelbtas mokslinėse publikacijose, doktoranto disertacijos darbe.

8. Su kuo galėčiau susisiekti, jei norėčiau pranešti apie tyrimą?

Jei nerimaujate dėl šio tyrimo aspektų, susisiekite su Aliaksei Kolesau, el. paštas: aliaksei.kolesau@vilniustech.lt, tel. (8 5) 274 4826. Sprendimas dėl Jūsų kreipimosi bus priimtas ir apie tai būsite informuotas per 30 darbo dienų.

9. Duomenų apsauga

Vilniaus Gedimino technikos universitetas yra duomenų valdytojas vgtu@vilniustech.lt, todėl tyrimui pateikti Jūsų asmeniniai duomenys bus valdomi institucijoje.

Vilniaus Gedimino technikos universitetas tvarkys Jūsų asmens duomenis anskčiau nurodyto tyrimo tikslais. Institucijos atliekami tyrimai vykdomi moksliniam tyrimui Balso aktyvavimo sistemų efektyvumo gerinimas naudojant mašininio mokymosi metodus.

Informacija apie teises į Jūsų asmens duomenis pateikiama Duomenų subjekto teisių įgyvendinimo Vilniaus Gedimino technikos universitete taisyklėse, nuoroda: https://vilniustech.lt/files/3013/150/7/8_0/Duomen%C5%B3%20subjekto%20teisi%C5%B3%20%C4%AFgyvendinimo%20taisykl%C4%97s.pdf

Asmens duomenų pareigūnas; el. pašto adresas: dap@vilniustech.lt; adresas korespondencijai: Saulėtekio al. 11, LT-10223 Vilnius. Skundas dėl asmens duomenų tvarkymo gali būti teikiamas Vilniaus Gedimino technikos universitetui ir el. paštu vgtu@vilniustech.lt , el. pašto adresas: dap@vilniustech.lt , adresas korespondencijai: Saulėtekio al. 11, LT-10223 Vilnius, Lietuvos Respublikos akademinės etikos ir procedūrų kontrolieriaus tarnybos el. pašto adresas info@etikostarnyba.lt , Valstybinės duomenų apsaugos inspekcijos el. pašto adresas ada@ada.lt.

10. Kontaktinė informacija

Jei norite iš anksto aptarti tyrimą (arba jei turėsite klausimų po tyrimo), susisiekite:

Aliaksei Kolesau
Vilniaus Gedimino technikos universitetas
Saulėtekio al. 11, LT-10223 Vilnius
aliaksei.kolesau@vilniustech.lt
(8 5) 274 4826

Gavau
Parašas
Data

Annex C. Informed Consent Form

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
Fundamentinių mokslų fakultetas

Dalyvio vardas, pavardė:
Dalyvio amžius:
Tyrėjo vardas, pavardė ir statusas: Aliaksei Kolesau, doktorantas
Tyrėjo institucija: Vilniaus Gedimino technikos universitetas
Tyrėjo tel. nr.: (8 5) 274 4826
Tyrėjo el. paštas: aliaksei.kolesau@vilniustech.lt

Nr		Jei sutinkate, pažymėkite varnelę	Jei nesutinkate, pažymėkite varnelę
1	Aš patvirtinu, kad perskaičiau ir suprantu minėto projekto / mokslinio tyrimo **Balso aktyvavimo sistemų efektyvumo gerinimas naudojant mašininio mokymosi metodus** informacinį lapą. Turėjau galimybę susipažinti su informacija, užduoti klausimus ir į juos gauti atsakymus.		
2	Aš esu informuotas, kad mano dalyvavimas yra savanoriškas ir kad aš galiu iš tyrimo pasitraukti bet kuriuo metu, nenurodydamas priežasties, nepatirdamas jokių neigiamų padarinių ar negaudamas baudų.		
3	Aš esu informuotas, kad tyrimo metu surinktus duomenis gali peržiūrėti įgalioti asmenys, nepriklausantys tyrėjų grupei (duomenų apsaugos pareigūnui, Lietuvos Respublikos Akademinės etikos ir procedūrų kontrolieriaus tarnybai, Valstybinei duomenų apsaugos inspekcijai, teismui).		
4	Aš esu informuotas, kas turės prieigą prie mano pateiktų asmens duomenų, kaip duomenys bus saugomi ir kas bus su duomenimis pasibaigus projektui.		

Tęsinys kitame puslapyje

6	Aš esu informuotas, kur kreiptis dėl tyrimo.		
7	Aš sutinku, kad būtų daromas garso įrašas.		
8	Aš esu informuotas, kaip garso įrašai bus naudojami apibendrinant tyrimų rezultatus.		
9	Aš sutinku, kad mano pasisakymai būtų cituojami tik nenurodant mano asmens duomenų.		
10	Aš sutinku dalyvauti tyrime		
11	Sutinku, kad šiame tyrime surinkti duomenys būtų teikiami tyrėjams, net ir tiems, kurie dirba už ES ribų, ir būtų naudojami kituose moksliniuose tyrimuose. Aš suprantu, kad visi duomenys bus visiškai anonimizuoti ir nebus galimybės nustatyti mano tapatybės.		

Dalyvio vardas, pavardė	Data	Parašas
Atsakingo asmens vardas, pavardė	Data	Parašas

Aliaksei KOLESAU

IMPROVING THE EFFECTIVENESS
OF VOICE ACTIVATION SYSTEMS
WITH MACHINE LEARNING METHODS

Doctoral Dissertation

Technological Sciences,
Informatics Engineering (T 007)

BALSO AKTYVAVIMO SISTEMŲ
EFEKTYVUMO GERINIMAS NAUDOJANT
MAŠININIO MOKYMOSI METODUS

Daktaro disertacija

Technologijos mokslai,
informatikos inžinerija (T 007)

Anglų kalbos redaktorė Jūratė Griškėnaitė
Lietuvių kalbos redaktorė Dalia Markevičiūtė

2022 07 29. 12 5 sp. l. Tiražas 20 egz.
Leidinio el. versija https://doi.org/10.20334/2022-033-M
Vilniaus Gedimino technikos universitetas
Saulėtekio al. 11, 10223 Vilnius
Spausdino UAB „Ciklonas",
Žirmūnų g. 68, 09124 Vilnius