Master's Thesis

Automatic Response Detection and Localization for a Clinical Picture Naming Test Using Deep Learning

Oleksii Zhelo

Examiners: Dr. Michael Tangermann
Prof. Dr. Joschka Boedecker
Adviser: David Hübner

Albert-Ludwigs-University Freiburg
Faculty of Engineering
Department of Computer Science
Brain State Decoding Lab

February 01st, 2019

Writing Period

01. 08. 2018 – 01. 02. 2019

Examiners

Dr. Michael Tangermann and Prof. Dr. Joschka Boedecker

Adviser

David Hübner

Declaration

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare, that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Place, Date	Signature

Abstract

This thesis presents a machine learning system developed to assist in the evaluation of naming test data produced by auditory aphasia patients undergoing rehabilitation. The assessment of treatment progress is performed based on the patient's ability to name the objects from a standardized picture set containing 233 entries selected to elicit responses from a fixed vocabulary of 230 words. As the naming test is repeated several times during the treatment, the amount of time required for manual processing of the data is very high, which calls for an automated solution to speed up the manual rating process. Since a notable proportion of rating time is spent finding the response words in naming test session recordings, we consider an automatic response detection and localization module, which would be integrated into an existing rating application to provide response location suggestions to the user.

To develop such a module, a collection of speech recordings from nine aphasic patients and four healthy controls, collected as a part of an experimental BCI-based aphasia treatment performed at the Brain State Decoding Lab lead by Dr. Michael Tangermann, supplemented by additional data from external datasets, was used to train a keyword spotting system. Variable-length audio data segments representing different words are converted to fixed-length embeddings with a recurrent neural network trained to achieve low distance between same-word-segment embeddings and large distances to embeddings representing other words. The trained network is then used to find sub-segments of novel recordings best corresponding to expected target words, achieving automatic detection and temporal localization of positive responses to picture prompts.

Contents

List of Figures

List of Tables

1 Introduction

Naming tests are widely used in assessment of the word retrieval ability in patients with various speech and language disorders. This type of test typically requires the participant to verbally name a sequence of commonplace objects depicted on a set of pictures preselected for recognizability, name consensus, and other criteria. The participant's performance is then evaluated based on test session recordings, which enables assessment of metrics such as reaction time until a correct response, duration of the word pronunciation, and the quality of said pronunciation.

Repeated or large-scale application of such tests presents a challenge for detailed evaluation of the results, as the assessment tends to require at least the same amount of time as the test session itself due to the need to review each recording, locate the response, and then calculate the desired metrics. While the complete recordings for each named picture can be quite long (up to 20 s in the data used in this work, with the mean duration of 4.03 s), the segment of interest containing the actual object name is much shorter, generally between 0.3 s to 1 s. As only a small part of these segments can be located trivially, such as by visual inspection of the audio waveform or with voice activity detection (VAD) approaches, a significant reduction in working hours required for rating could be achieved by automatizing their detection.

In this work we aim to accomplish the following goals: 1) develop a lightweight and non-resource-intensive system for automatic naming test response detection which can be distributed alongside an existing solution for naming test evaluation, 2) lay the foundation for future work towards complete automation of the rating process, including speech quality analysis.

To accomplish these goals, the following tasks need to be solved in this work: 1) review of existing speech recognition approaches and selection of the method most suitable for response detection and future speech quality analysis work, 2) collection and preparation of the training data, 3) definition of performance metrics to evaluate the proposed system, 4) development and evaluation of a response detection approach.

1.1 Naming tests for aphasia patients

This work focuses on the specific test type used to monitor the progress of aphasia treatment in a currently ongoing study performed at the Brain State Decoding Lab. Aphasia refers to a communication disorder caused by damage in specific brain regions responsible for comprehension and formulation of language, as well as speech production. Depending on the specific symptoms, aphasia can be broadly classified into two types [1]:

- **Non-fluent** aphasia is characterized by slow, halting, effortful speech, often limited to very short disconnected expressions, with multiple disfluencies and restarts.

- In **fluent** aphasia the ability to grasp the meaning of spoken and written language is damaged, while the capacity to produce grammatically connected sentences, as well as articulation, intonation and rate of speech, are not strongly affected. Individuals with fluent aphasia often use wrong or non-existing words and produce uninterpretable speech.

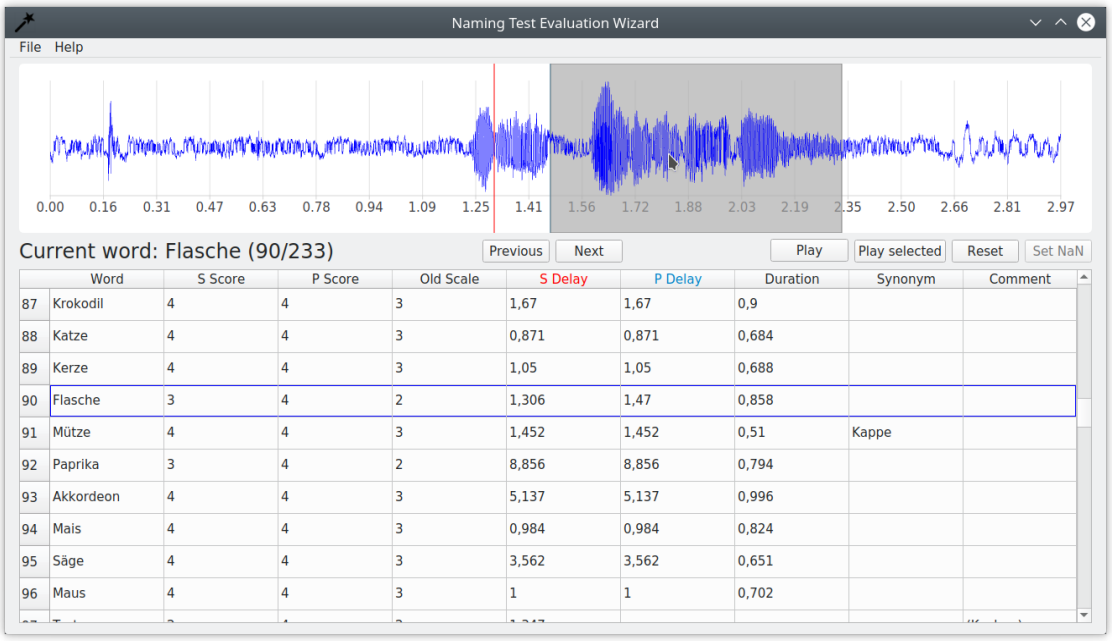
In both types word-finding ability and motor speech production can also be affected.

Overall, aphasic speech presents the following challenges for automatic speech recognition (ASR), which need to be accounted for in the design of our proposed solution:

- The amount of data available for training is very low, compared to the typical abundance of richly transcribed recordings of healthy speech.

- Aphasic speech properties need to be considered in the recognition system structure. For instance, ASR systems typically include a language model component, which is used to help determine the most likely word sequence corresponding to a particular speech recording. Such models are typically implemented using the n-gram approach, in which the next word x_i is predicted based on $n-1$ previous words, modeled with a probability $P(x_i \mid x_{i-1}, \ldots, x_{i-(n-1)})$. This probability is then estimated from a large text or speech corpus. As previously described, aphasic speech can significantly differ from the norm, either with omissions of connecting words, such as verbs, or with additions of unsuitable/non-existing words. Consequently, a language model trained on normal speech data is likely to poorly describe aphasic speech.

1.2 Naming test evaluation



	Word	S Score	P Score	Old Scale	S Delay	P Delay	Duration	Synonym	Comment
87	Krokodil	4	4	3	1,67	1,67	0,9		
88	Katze	4	4	3	0,871	0,871	0,684		
89	Kerze	4	4	3	1,05	1,05	0,688		
90	Flasche	3	4	2	1,306	1,47	0,858		
91	Mütze	4	4	3	1,452	1,452	0,51	Kappe	
92	Paprika	3	4	2	8,856	8,856	0,794		
93	Akkordeon	4	4	3	5,137	5,137	0,996		
94	Mais	4	4	3	0,984	0,984	0,824		
95	Säge	4	4	3	3,562	3,562	0,651		
96	Maus	4	4	3	1	1	0,702		

Figure 1: Naming test evaluation application.

The naming test data this work focuses on is evaluated using the application depicted on Figure 1, which was developed by the author as part of previous work in a student research assistant capacity at the Brain State Decoding Lab.

The data is processed on a per-session basis. For each session, the rater's task is to review individual recordings corresponding to picture prompts and evaluate the patient's responses on a semantic and phonological scale. The semantic scoring aspect is not considered in this work. The phonological ratings are performed on a five-point scale as follows:

- A score of 0 corresponds to an unrecognizable target word.
- A score of 1 is given if at least one third of the target word phonemes is recognizable in the best response attempt.
- A score of 2 is given for two thirds of recognizable target word phonemes.
- A score of 3 is given for a fully correct word pronunciation after one or more failed attempts.

- A score of 4 corresponds to a fully correct word pronunciation without complications.

Before this rating can be given, the best response attempt needs to be located in the recording. Its location is manually marked with a phonological delay value, corresponding to word production start, and the pronunciation duration value, inferred from the word duration window selected by the rater (indicated by a gray rectangle over the audio waveform plot on Figure 1).

The patient can also respond with a synonym instead of the expected target word, which is documented by entering the used synonym in the corresponding column. This work, however, excludes synonyms from the recognition vocabulary due to their rarity and to simplify the response detection system.

2 Related work

We have consulted several previous works investigating the use of automatic speech recognition for aphasic speech. All of them rely on Hidden Markov Model (HMM) based approaches, which we discuss in subsection 3.1.1.

Abad et al. [2] adapt an ASR system trained exclusively on healthy speech data to perform response spotting during word naming exercises with the goal of providing online feedback without a speech therapist. Their approach does not perform acoustic model retraining or adaptation and is intended to target persons with mild aphasia, suffering mostly from difficulties with word-finding.

Le et al. [3] develop an approach for automatic analysis of aphasic speech quality based on manually transcribed recordings of picture naming sessions. They extract transcript and acoustic features from the data and train classifiers to match speech quality scores based on human judgment. A followup work [4] develops a continuous speech recognition system trained on a healthy speech corpus and adapted by retraining on a smaller aphasic speech dataset. The resulting system is used to automatically produce the transcripts necessary for speech quality analysis. Their work exploits the knowledge of speech prompts the patients are expected to follow in their responses, constraining the search space of potentially produced sentences, which is a simplification not available for our task.

Fraser et al. [5] investigate aphasia diagnosis based on transcriptions of spoken speech using lexical and acoustic features. Their work highlights the importance of considering aphasic speech properties in the speech recognition pipeline.

3 Methods

As mentioned in Chapter 1, the task of automatic naming test evaluation can be approached in two stages: 1) locating the patient's responses, 2) analyzing the quality of the detected responses. In this work we focus on the first stage, however the developed solution should allow a natural progression towards solving the complete test evaluation task.

The first stage corresponds to the well-known problem of keyword spotting (KWS). There are several ways this task has been typically approached in the literature, depending on the size of the term vocabulary to be detected, availability of training data, and the specific type of data available. This chapter will begin with a short overview of the standard KWS techniques and selection of the approach most suitable for naming test evaluation. The rest of the chapter will describe the details of the selected approach and the performance metrics for its evaluation.

3.1 Approach selection

The predominant approach to automatic speech recognition is searching for the most likely sequence of words W corresponding to the acoustic observations O:

$$W^* = \arg\max_{W \in \mathscr{L}} P(W \mid O), \tag{1}$$

where $\mathscr{L}$ is the set of all possible words [6]. For the keyword spotting task this set commonly includes a "background word" which corresponds to all non-keyword speech, and the actual terms to be detected. Another option is to explicitly model as many words as possible and simply look for keywords among the detected words. Depending on this decision and on the model choice for $P(W \mid O)$ many different ASR approaches have been developed, the main types of which we will consider in the context of naming test evaluation in this section.

3.1.1 HMM-based systems

We can use Bayes' rule to transform Equation 1 as

$$W^* = \arg\max_{W \in \mathscr{L}} \frac{P(O \mid W)P(W)}{P(O)} = \arg\max_{W \in \mathscr{L}} P(O \mid W)P(W), \tag{2}$$

where $P(O)$ is ignored as it is constant for each potential word sequence. In this interpretation $P(W)$ gives the prior probability of a word sequence and is implemented by a **language model**, and the observation likelihood $P(O \mid W)$ by an **acoustic model**.

Hidden Markov Models (HMMs) are one of the most successful approaches for building acoustic models. An HMM is characterized by a set of states $Q = \{q_1, \ldots, q_N\}$, a transition probability matrix $A = \{a_{ij}, i = 1, \ldots, N, j = 1, \ldots, N\}$ such that $\forall i \ \sum_{j=1}^{N} a_{ij} = 1$, and a set of observation likelihoods $b_i(o)$ giving the probability of observation o being generated in state q_i. The process of speech production is interpreted as building a trajectory consisting of hidden states, where the state determines the properties of the observations being generated. Additionally, the Markov assumption that the probability of the current state depends only on the previous state is made, and the observations are assumed to be independent of each other and dependent only on the current state.

To reduce the number of parameters to be estimated, the HMM states are typically taken to be the basic sound units – *phones* instead of words, which allows to model an arbitrary number of words using a fixed state set. Individual word models are then built from phone models by concatenating an appropriate sequence of phones determined by a pronunciation lexicon.

The HMM transition probabilities and observation likelihoods can be estimated using the forward-backward algorithm [7], after which the trained system can be used to search the space of all possible state sequence hypotheses and find the most likely one under the model, typically employing Viterbi search. The search is kept tractable by pruning unlikely hypotheses in a complex decoder scheme, an overview of approaches to which can be found in [8].

There are several ways the described machinery could be employed for the KWS task:

- An HMM system can be trained for large vocabulary continuous speech recognition (LVCSR) on a large speech corpus and employed to transcribe the audio signal into text with associated timestamps on a word level. This approach

would require a significant amount of training data, as well a large pronunciation lexicon and a comprehensive language model. After the transcription, text-based search methods could be used to locate the keywords.

- Another approach performs a similar transcription, but on a phoneme level [9]. The output of such a system is a string or lattice of phonemes, which can be searched for phoneme sequences most closely matching the sought keywords, according to a pronunciation dictionary. This approach is more flexible, as it not restricted to a fixed keyword set and allows finding inexact word matches, which is relevant for atypical speech we can expect from aphasic patients. This and the previous approaches can also be combined to produce both word and phonetic transcriptions [10].

- Instead of using a large vocabulary, only the expected keywords can be modeled explicitly, and a general speech model can be used to represent all non-keyword speech. This architecture is called a keyword-filler HMM [11, 12]. Keyword detection is performed by running Viterbi search and checking whether the most probable state sequence contains the keyword states. Subsequent text search is not necessary here, as the audio stream will be directly tagged as belonging to specific keywords or to background speech.

The main argument against choosing one of these approaches for our task is their high complexity and the associated implementation difficulty, computationally intensive training and application. The assumptions necessary to make this approach tractable are known to be violated in real speech data, leading to complex modeling to mitigate the negative effects of these violations. An important example of this is the widespread usage of context-dependent phones to help uphold the observation independence assumption, which leads to quadratic or cubic growth of the state space and requires language-specific knowledge of possible phone sequences [6].

Building of word models for aphasic speech would be complicated by the pronunciation mistakes common for persons with aphasia, and compilation of a corresponding pronunciation lexicon would require extensive domain knowledge and large amounts of data.

The transcription-based models rely on a language model for high recognition accuracy, however accurate estimation of word- or phoneme-level aphasic language models is not possible due to a lack of data. While the keyword-filler approach is less complex and does not necessitate language models, building a good background

speech model requires training on additional non-keyword speech [11], transcribed examples of which are lacking in our aphasic speech dataset. Additionally, only the phonetic transcription approach provides a clear way to extend the system for speech quality assessment (similar to what was done in [4]).

3.1.2 Discriminative KWS

Unlike the previously described approaches, discriminative systems are based on modeling the word sequence posterior $P(W \mid O)$ directly. Since in the context of KWS we are not necessarily interested in recovering the entire sequence of words corresponding to the speech signal, but only need to find separate occurrences of keywords in the speech stream, the problem statement is typically simplified to making a prediction about keyword presence from individual audio stream segments.

A notable example of such systems was developed by Chen et al. [13] in order to address keyword spotting on mobile devices, which lack the computational capacity required for HMM decoding. Their proposed system used a neural network trained to directly predict the keywords (or their absence) based on small segments of audio input (300 past and 100 future speech milliseconds), removing the need for time- and resource-consuming decoding and showing performance levels superior to the previous state-of-the-art approaches.

The simplicity and speed of this approach make it very attractive for the response detection part of naming test evaluation, however it is not clear how to implement an extension to response quality assessment in this setting, which is the main reason we do not adopt this technique for our task. Moreover, future changes of the recognition vocabulary would require a complete retraining of the system, and incorporation of new data or adaptation to new patients would also lead to retraining or fine-tuning, which may not be optimal for a practical application.

3.1.3 Template matching

Also called "query-by-example search", template matching KWS methods are based on computing distances between keyword audio examples and all possible segments of the test signal [14]. The keyword is considered to be detected in a segment of audio signal when the computed distance is below a predefined threshold.

Early approaches used dynamic time warping (DTW) to compute an alignment distance between speech segments [15]. Disadvantages of using DTW include poor

performance due to speaker-dependent speech variability, recording conditions mismatch between the template audio and the test signal, and high computational costs (due to DTW's quadratic time and space complexity).

Recent works propose an alternative to DTW in which arbitrary-length speech segments are embedded in a fixed-dimensional space constructed specifically so that same-word segments have similar embeddings [16, 17, 18, 19, 20]. Speech segments can then be easily compared based on a standard metric such as Euclidean or cosine distance. Besides simplifying distance calculations, neural-network-based embeddings have the potential to reduce speaker dependency and environmental influences by learning features invariant to these factors. The usage of acoustic word embeddings allowed the development of faster and more accurate template matching KWS systems, such as the work of Settle et al. [21].

Template matching coupled with neural-network-based embeddings seems to be the most promising approach for aphasic speech naming test evaluation, as it is both simpler than HMM-based approaches and more flexible than classification-based discriminative KWS. Using templates allows to adapt the recognition system to new patients by incorporating their responses as reference examples after a single naming test session, without retraining the whole system. Acoustic word embeddings also enable an extension towards response quality assessment, potential approaches to which will be described in Chapter 6.

The next two sections will discuss acoustic word embedding training and describe the response detection system built on top of them.

3.2 Acoustic word embeddings

The task of learning fixed-size vector representations of variable-sized audio segments corresponding to words can be formalized as finding a function $f : \mathcal{X} \to \mathbb{R}^m$, where $\mathcal{X} = \{X = \{x_t\}_{t=1}^T\}$ is a set of all arbitrary-length sequences of acoustic features representing audio segments. In this work we represent the function f with a recurrent neural network (RNN), which allows natural processing of variable-sized input, and we will refer to the output vector representations as acoustic word embeddings.

The embedding models are trained on pre-segmented spoken words, and the approaches used in their training are described in the next two subsections.

3.2.1 Similarity-based learning

In order to make acoustic word embeddings useful, we would like to induce a similarity relation between them: representations of sound segments corresponding to the same word should be similar to each other and dissimilar to representations of all other sounds. A natural way to quantify similarity of vectors in $\mathbb{R}^m$ is via the distance between them, so that closer vectors are considered to be more similar. This similarity relation is extended to sound segments through the embedding function f, so that the Euclidean distance between two segments can be found as $D_{ij} = \|f(X_i) - f(X_j)\|$, and cosine distance as $D_{ij}^{cos} = 1 - \cos(f(X_i), f(X_j))$.

Similarity-based learning follows the idea of Siamese networks [22], which accept pairs of input vectors and minimize or maximize the distance between them. Consequently, we will refer to networks trained in this manner as "Siamese".

Triplet loss

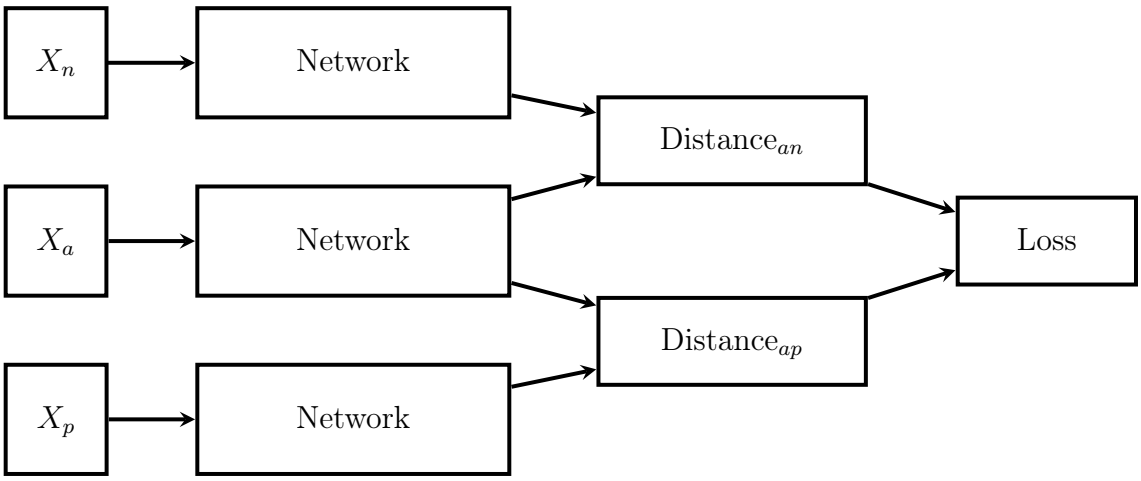


Figure 2: Basic triplet loss training scheme.

One of the best-performing approaches for learning similarity-aware embeddings is triplet-loss-based training (as benchmarked in [23]), successfully used across a variety of different tasks. For acoustic embeddings, triplet loss usage was first proposed by Kamper et al. [19], while in computer vision community it was popularized by Schroff et al. [24] who used it for large-scale face recognition in the FaceNet framework.

Triplet loss is defined over samples composed of a random *anchor* example, a *positive* example of the same class, and a *negative* example belonging to a different

class, with distances between them indicated as D_{ap} and D_{an}:

$$\ell^{\text{triplet}}(X_a, X_p, X_n) = \Big[\alpha + D_{ap}^2 - D_{an}^2\Big]_+, \tag{3}$$

where α is a parameter giving the desired separation margin between positive and negative pair distances, and the embeddings are normalized to unit length for training stability, following [24].

Training on triplets emerged as a more flexible alternative to the pair-based contrastive loss developed by Hadsell et al. [25], given by

$$\ell^{\text{contrast}}(X_i, X_j) = y_{ij} D_{ij}^2 + (1 - y_{ij})\Big[\alpha - D_{ij}\Big]_+^2, \tag{4}$$

where $y_{ij} = 1$ if X_i and X_j belong to the same class, and $y_{ij} = 0$ otherwise. This loss forces all same-class examples to be close, separated from all negative examples by a fixed distance α. Adding the third anchor example allows to define the separation goal relatively to distances between same-class pairs, only forcing different examples to be further away than similar ones. As argued in [24], this allows each class to form a manifold instead of being forced to collapse into a single point, while still enforcing discriminability to other classes.

We experimentally compare two variants of the triplet loss:

1. The variant of Settle and Livescu [20], which replaces squared Euclidean distance with cosine distance and does not employ embedding normalization. Even though the distance metric is different, this is equivalent to using Equation 3 with normalization, as $D_{ij}^2 = 2 \times D_{ij}^{cos}$ for normalized vectors.

2. The same loss as Equation 3, but without squaring the distance. As suggested in the analysis of Wu et al. [23], removing the square should help with the problem of the model collapsing to all inputs being represented with identical embeddings, which tends to happen with the previous loss formulation.

Minimizing this loss over all triplets that can be built from the training set is infeasible, as their number grows with $O(n^3)$, which motivates the development of many triplet selection heuristics (commonly referred to as "triplet mining"). As argued in [26], Siamese networks quickly learn to correctly separate most "trivial" examples, which are typically very numerous (eg. it's easy to learn that people in different clothes are different in the task of person identification), which means that

sampling triplets at random is unlikely to produce many good examples. Striving to find the triplets which violate the margin constraint in Equation 3 the most, on the other hand, is not only very computationally expensive, but can also lead to overfitting to outlier examples and hurt the model's generalization capability. A common approach is then to find a middle ground between these extremes, such as the "batch-hard" or "semi-hard" approach of [24], where for each anchor-positive pair a negative example is found such that

$$n^\star = \underset{n:D(a,n)>D(a,p)}{\arg\min} D_{an},$$

from the current batch. This gives a negative example that is minimally further away from the anchor than the positive example, which is still useful as it is still approximately as close as the positive example.

We select an equal amount of anchor-positive pairs for each word randomly and sample 10 negative examples from different classes for each pair, completing the triplet with the negative example that leads to the largest loss, similar to the approach taken by Settle et al. in [21].

Adaptive margin loss

The pair-based adaptive margin loss was proposed by Wu et al. [23] to combine the flexibility of triplet-based training and the higher computational efficiency of the loss given by Equation 4. It is defined as

$$\ell^{\text{margin}}(X_i, X_j) := \Big[\alpha + y_{ij}(D_{ij} - \beta)\Big]_+, \tag{5}$$

where α gives the margin of separation between positive and negative examples, β is a learnable boundary relative to which the margin is enforced, and $y_{ij} = \{-1, 1\}$ for negative and positive pairs correspondingly. Here the β parameter can be learned according to its gradient on per-class or per-example basis.

Wu et al. [23] also propose a new strategy for selecting training pairs, based on uniformly sampling them according to between-pair distance. As the embeddings are constrained to an m-dimensional unit sphere because of length normalization, the distribution of their pairwise distances follows

$$q(d) \propto d^{m-2}\Big[1 - \tfrac{1}{4}d^2\Big]^{\frac{m-3}{2}}. \tag{6}$$

A derivation is available at [27]. Wu et al. propose to sample negative pairs (a, n) given a reference example a according to

$$\Pr\Big(n^\star = n|a\Big) \propto \min\Big(\lambda, q^{-1}(D_{an})\Big),$$

where the sampling is done from all negative examples in the batch. The cutoff with λ is done to avoid noisy samples. Sampling in this way should provide the training with a balanced combination of examples, without the pairs being either always very hard or very simple, which should stabilize learning and improve convergence, while avoiding the computational cost of triplet mining.

3.2.2 Fully supervised learning

In this setting, instead of training a neural network model to represent a mapping $f : \mathcal{X} \to \mathbb{R}^m$ from audio input to fixed-size embeddings directly, we append a softmax layer with one output unit per each distinct word to the base network and perform training with the standard cross-entropy loss. The complete model is then estimating a posterior distribution over words given the audio segments corresponding to them. As argued in [18], the layer before the softmax layer of a model trained in this manner contains a good representation of the input audio data, and can therefore be considered as implementing an acoustic embedding. In the space of this embedding the words that sound similarly can be expected to have similar representations.

There are several drawbacks to this approach. First, training a good classifier typically requires a large number of examples for each class, while the available data has very few (less than five) examples of some of the words, and many more (over 500) examples of more commonly occurring words. This also leads to the problem of unbalanced data, which is likewise problematic for classifier training. Second, the distinct words used as classes in the classifier need to be known in advance. While this work focuses on a fixed keyword detection vocabulary, it does not currently include synonyms patients sometimes use in place of the expected target words. Extending the vocabulary would require a full retraining of the system.

In this work the classifier-based approach is important in two aspects: 1) as a benchmark for between-word classification accuracy, 2) as pre-training for initialization of the networks trained with cosine-distance-based triplet loss as was done in [20], which otherwise fail to converge.

3.3 Response detection

As described in subsection 3.1.3, keyword detection is performed via comparisons of test signal segments to reference keyword examples. Using templates allows us to exploit the fact that we know the expected response for each test item, so we can use the correct word examples for simpler and faster matching instead of performing comparisons against all possible keywords. The previously described acoustic word embeddings play a major role in the detection system, as we perform all comparisons based on distances between embeddings of the test and reference audio segments.

Following Levin et al. [28], a set of candidate sub-segments can be found by 1) using voice activity detection (VAD) to find regions likely to contain speech, 2) splitting each VAD-detected region into overlapping segments from some minimum to some maximum duration. Afterwards, the sub-segments are mapped to embeddings and evaluated based on their distances to keyword templates to find the most likely keyword location. Here we adopt a very similar approach following this basic scheme.

We investigate two approaches to segment the input recording into regions of interest:

1. In **VAD-based segmentation** we use an open-source VAD solution available as part of the WebRTC project [29] (with the source code available at [30]) to produce a list of potentially speech-containing regions. As the patients sometimes make long pauses while pronouncing the response word, voice activity detection can split the response into two or more separate regions. We counteract this by adding additional segments to the output list by concatenating consecutive regions if the gap between them is less than a second.

2. In **sliding window segmentation** we cut out a sequence of sub-segments from the whole input signal by sliding a window over it. The window length is defined as half of the expected keyword mean duration (mean calculated from the training data), and the window is shifted in increments of 50 ms. The resulting segments are mapped to embeddings, which are then used to calculate the average distance to all training set examples of the expected keyword. This produces a distance curve like one depicted on Figure 3, where the minimum corresponds to a segment assumed to coincide the best with the sought keyword beginning. We expand this segment to the expected keyword maximum duration (calculated from the training data) plus an additional 500 ms and return it as the single potential region of interest for further processing.

After the candidate regions are determined, we segment them further to narrow down the potential keyword location. As generating all possible sub-segments of the found regions would require too much following computation, we simplify the splitting step on an assumption that the beginning of each region is more likely to correspond to the correct keyword beginning, while the region end can contain external speech or unchaught noise. With this assumption only the endpoint of the potential keyword segment needs to be determined. Consequently, we split the region into sub-segments with the beginning fixed at the original start time, adding progressively more of the region in 10 ms increments (corresponding to cumulatively appending consecutive frames of the original region). These incremental sub-segments are then mapped to embeddings, which allows to compute a distance curve as depicted on Figure 4. This curve plots the average distance to all training set examples of the expected keyword for each sub-segment, with the sub-segment endpoint as the x-coordinate. The minimum of this curve corresponds to the sub-segment closest in average to the reference keyword examples, which we use as the best guess of the keyword location, with the associated average distance as the detection score.

The output of our detection process so far is a list of potential keyword detections, each containing a recording segment with a distance score. The final step is thresholding these detections based on their scores to find the most likely keyword location, or to reject them completely. As we have found that detections occurring earlier in the recording are more likely to be correct (mostly due to response repetitions), we perform thresholding with two thresholds: a lower one (more restrictive) for **immediate** detection acceptance, and a higher one for **exhaustive** search. In this scheme we iterate over potential detections in chronological order, accepting the first one with a distance score under the **immediate** threshold, or, if no detections have a low enough distance, accept the detection with the lowest score which is under the **exhaustive** search threshold. If all detection scores are too high, we assume the expected response was not produced.

The thresholding is performed in two modes:

1. With a sweep over all possible thresholds, where the **immediate** threshold goes from 0.1 to 0.4 in increments of 0.1 and stops growing, and the **exhaustive** threshold goes from 0.2 to 1.0 with same increments.

2. For embedding networks based on the adaptive margin loss, the same-different-class boundary parameter β learned for each class gives a natural threshold,

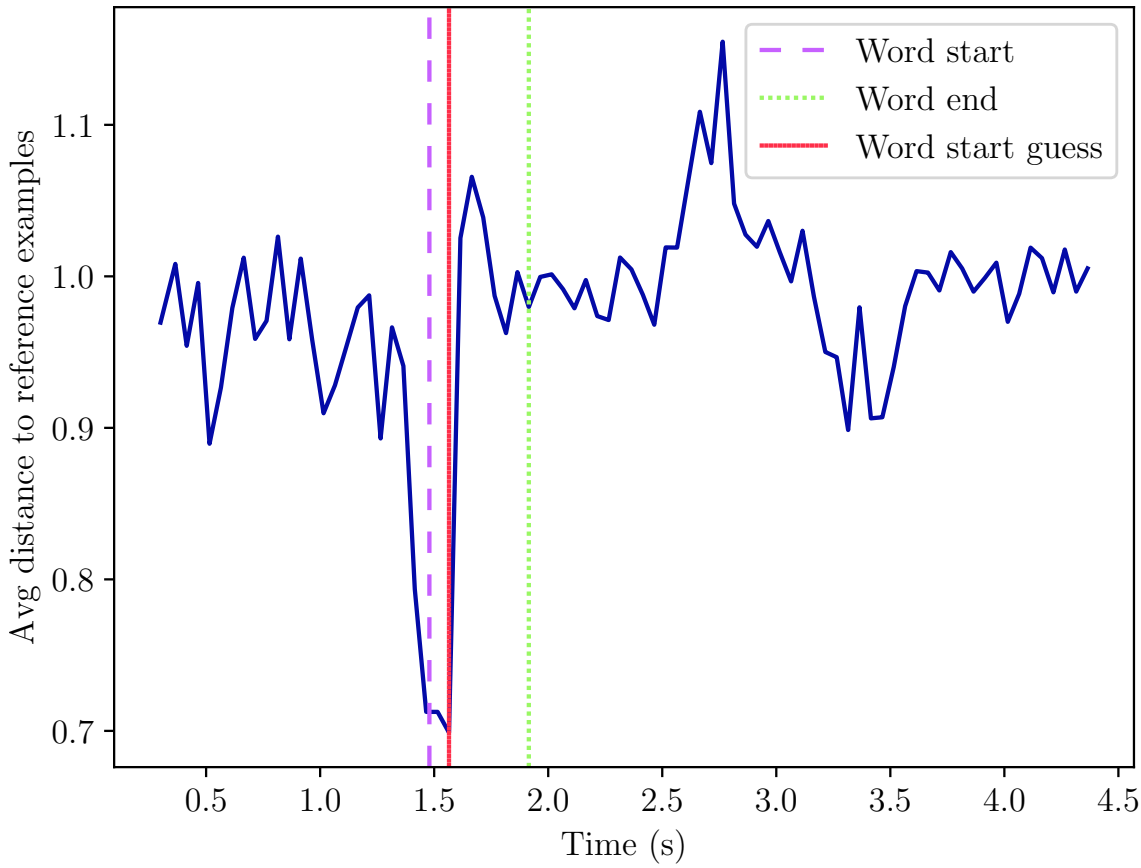


Figure 3: Moving window average distance to reference examples. Finding the minimum allows to determine the most likely word beginning point (marked with a red line).

combined with the separation parameter α. In this case we use the class-dependent boundary in this way: the **immediate** threshold is set to $\beta - \alpha$, and the **exhaustive** threshold is set to $\beta + \alpha$.

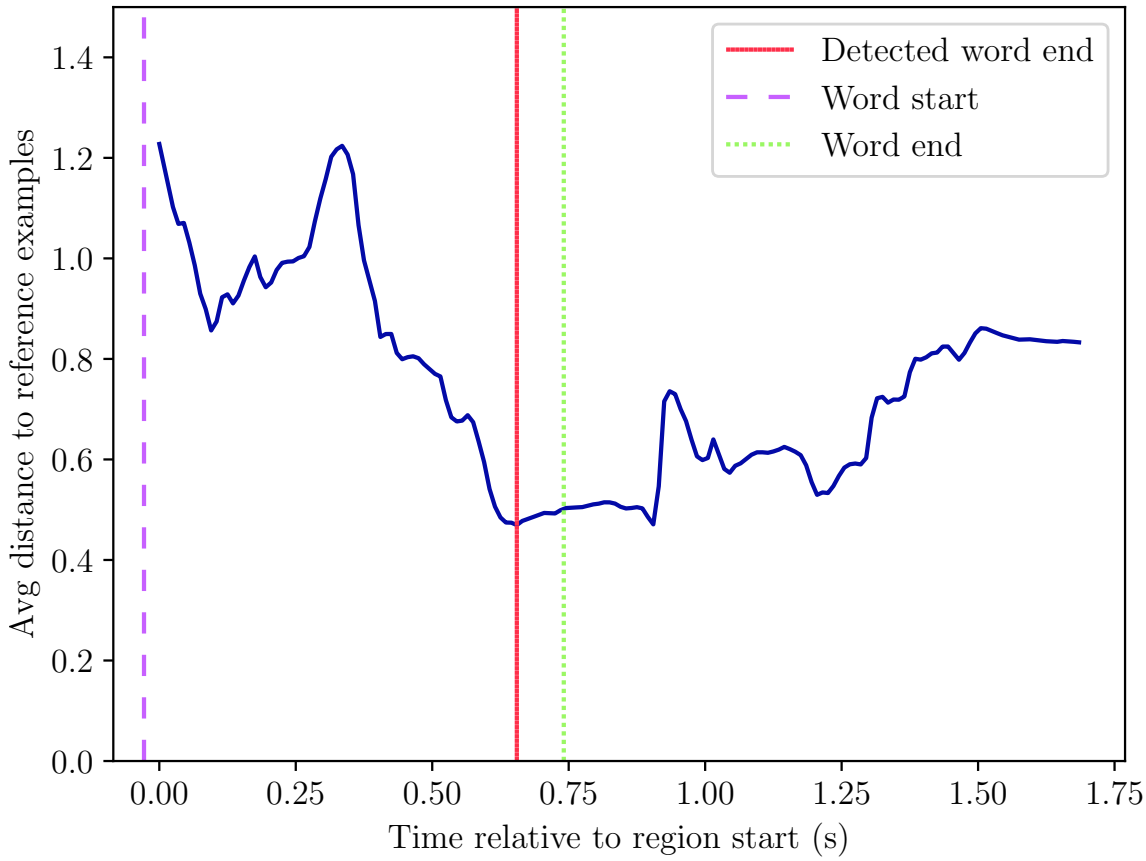


Figure 4: Framewise average distance to reference examples for a VAD-detected region. Using the minimum distance frame allows to determine the most likely word ending point (marked with a red line), as the sub-segment spanning the interval from the region start up to this frame is the closest to keyword templates.

3.4 Feature extraction

Speech has many sources of variation not connected to the message being communicated: two instances of the same sentence being spoken can result in two very dissimilar digital speech signals. This "external" variation contains information about different aspects of the speech production and transmission process, such as the specifics of the speaker's vocal tract configuration, their emotional state, the quality of the recording equipment, or the specific environmental conditions like background noise, distance to the microphone, etc. As the external information tends to adversely affect speech recognition performance, it is preferable to preprocess the raw audio input to discard it, while retaining as much of the message-relevant aspects of the

audio data as possible.

Another motivation to seek a transformation of the raw speech data into a feature representation lies in the low intrinsic information rate of speech, as described by Rabiner and Schafer in [31]: assuming 5 bits per symbol and a typical speaking rate of 10 symbols per second, the base information rate of a text message is approximately $50\,\mathrm{bit\,s^{-1}}$, while the data rate of the digitized speech waveform can range from $64\,000\,\mathrm{bit\,s^{-1}}$ up to $700\,000\,\mathrm{bit\,s^{-1}}$ and more. Consequently, it would be advantageous to obtain a feature representation of the sampled data which captures the underlying more slowly changing information content of speech and mitigates the effects introduced by the transmission and digitization process.

Based on these considerations, a multitude of feature extraction approaches were developed, one of the most commonly used of which is the mel-frequency cepstral coefficients (MFCC) [32]. While modern deep learning approaches generally tend to move away from manual feature engineering towards full end-to-end learning [33, 34], in the area of speech recognition learned features have only recently begun to catch up in performance compared to state-of-the-art handcrafted features, and lead to a comparatively more involved extraction process [35]. Consequently, following Chen et al. [13], we use mel-frequency log filterbank energies (LMFE), which are equivalent to MFCCs without the final DCT stage, as the neural network inputs.

LMFE

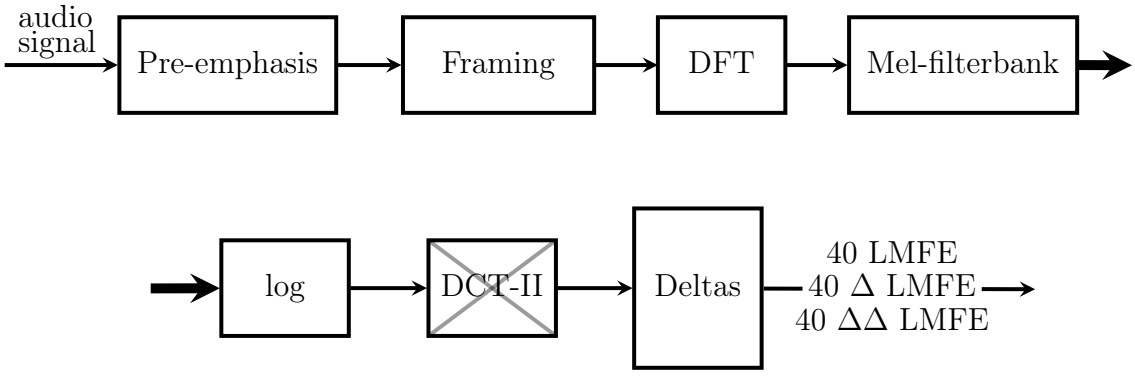


Figure 5: Extracting LMFE features from an audio signal.

The extraction of LMFE features as performed in this paper is summarized in Figure 5 and includes the following steps:

1. The **pre-emphasis** step is done with a filter in the form of

$$y[n] = x[n] - \alpha x[n-1],$$

where $x[n]$ is the input audio signal, and α is a parameter between 0.9 and 1. Pre-emphasizing the signal boosts the amount of energy in the higher frequencies, which is otherwise low in comparison to the lower frequencies.

2. Next, the speech signal is converted into a sequence of short **frames** by multiplying it with a window function as the window is slid along the time axis. The window needs to be short enough to assume that the signal is stationary within its length, typically 20 ms to 40 ms with a frame shift of 10 ms to reduce boundary artifacts and smooth out the changes between the frames.

3. In the next step each frame's power spectrum is computed using the **Discrete Fourier Transform (DFT)**, which is inspired by the frequency analysis performed as part of human hearing.

4. The next step summarizes the power spectrum's content using a series of **triangular filters** equally spaced in the **mel scale**, an example of which is depicted on Figure 6. The mel scale is designed so that sounds which are perceived to be equally distant in pitch are separated by the same number of mels. Several Hz to mel conversion formulas exist, this work uses the following one:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right).$$

The summarization is motivated by the fact that human ear cannot discern between closely spaced frequencies, with the frequency resolution dropping towards the higher end of the spectrum, hence the widening of the filters.

5. Taking the **log** of the filterbank energy values is also motivated by human hearing, which is less sensitive to changes in sound intensity at higher levels than at lower levels. Additionally, taking the logarithm reduces the overall influence of volume level variations, for instance due to changing distance to the microphone.

6. The **Type II Discrete Cosine Transform** step is the final part of standard

MFCC calculation, which we omit in our LMFE extraction pipeline (hence the crossing out of the corresponding block in Figure 5). One purpose of DCT is to decorrelate the features which was historically very useful for speech modeling with diagonal covariance matrices, but is less relevant for modern approaches. The second motivation behind DCT usage is dimensionality reduction, usually performed by discarding higher DCT output coefficients.

7. The final features can be augmented with first and second time derivatives, called **delta** and **delta-delta** features, to capture the dynamic properties of speech. While the RNN-based models used in this work should be able to capture dynamic information without this augmentation, using it still leads to a small performance increase.

Overall, we compute the features over speech signal windows of 25 ms with a window shift of 10 ms using a mel-filterbank with 40 filters and augment them with deltas and delta-deltas, leading to 120-dimensional feature vectors for each frame.

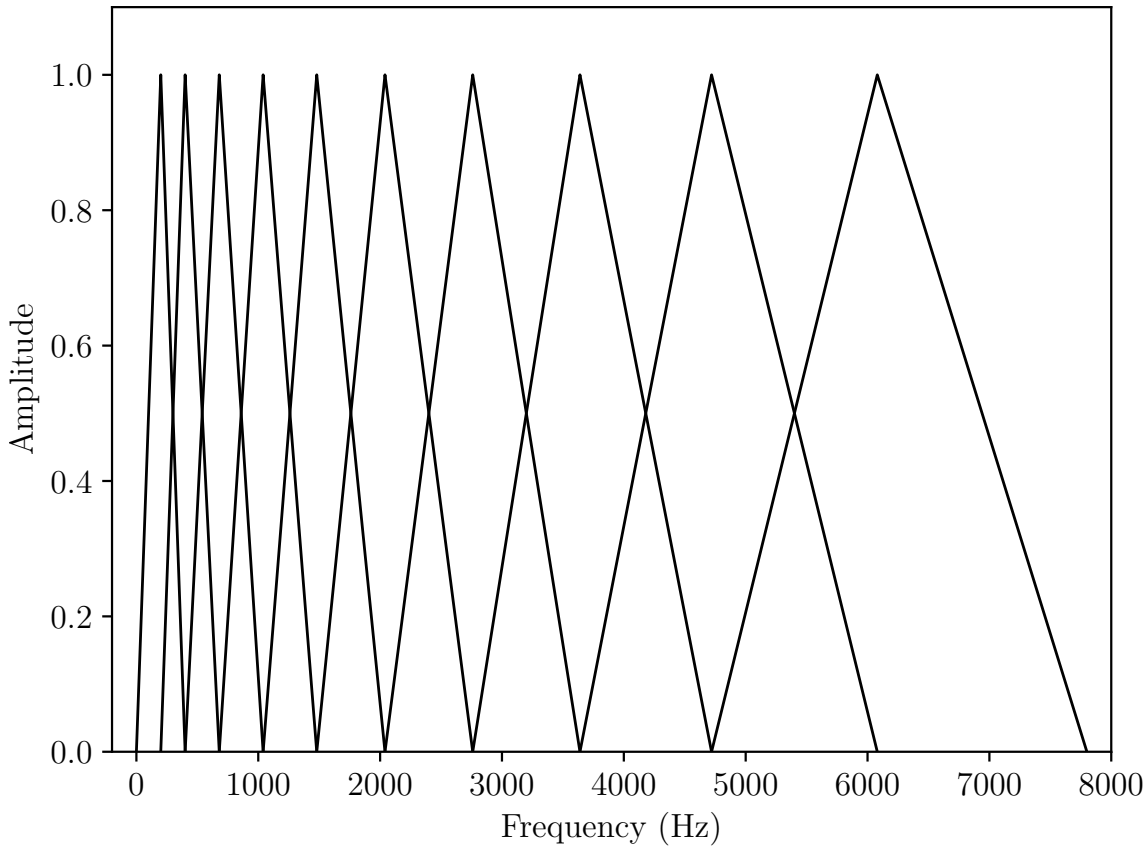


Figure 6: A mel-filterbank with 10 filters.

3.5 Performance evaluation

The assessment of the response detection pipeline is divided in two stages. In the first stage the learned acoustic embeddings are evaluated based on the similarity between same and different word pairs, as well as on the classification accuracy of a classifier built on top of the embeddings. The second stage evaluates the naming test response detection performance when using the best acoustic embeddings determined in the previous step.

3.5.1 Embedding quality

The embeddings are evaluated based on correctly segmented words, so here we only test their discriminative ability assuming perfect input. We evaluate whether the learned embeddings have the expected similarity relationship between same and different word pairs using the average precision (AP) metric, following Settle and Livescu [20]. Average precision is determined by calculating distances between all pairs of evaluation segment embeddings and averaging the precision values obtained when taking each of the distances as the decision threshold below which the segments are considered to be the same.

We additionally calculate the accuracy of a k-nearest neighbors classifier built on top of the embeddings: evaluation segments are mapped to embeddings and assigned to the majority class of the three nearest neighbors from the training set. This is done to compare the classification performance of select best embedding models trained with triplet or adaptive margin losses to the classifier network of subsection 3.2.2. We report the accuracy as a value between 0 and 1, unless specified otherwise.

3.5.2 Response detection

The response detection system is evaluated on complete recordings which were the source of test set word segments, including the recordings in which the patient failed to produce a good response. For each recording the system outputs a list of potential detections, each containing a corresponding segment and a distance-based score. As described in Section 3.3, the potential detections are thresholded based on their scores, and are either **rejected**, or a single detection is **accepted**. If the system did not return any detections for a recording, we will call this a **missing** detection.

Each recording from the patient data is associated with a manually performed

rating, which reports whether the word was correctly produced, and the locations of the word's beginning and end relative to the whole recording. The patient can also respond with a synonym of the word, which is also reported.

Comparing the detections with a human rating allows to calculate the precision and recall scores, where the comparison is done as follows:

- an **accepted** detection is counted as a **true positive** when the human rating reports the word as correctly produced, without synonym usage, and the automatically detected segment beginning and end are both not further than 200 ms from the corresponding manually reported word boundary
- an **accepted** detection is counted as a **false positive** when the human rating reports the word as not produced, or a synonym was used, or the automatically detected segment boundaries violate the closeness condition
- a **rejected** or **missing** detection is counted as a **true negative** when the human rating reports the word as not produced, or a synonym was used
- a **rejected** or **missing** detection is counted as a **false negative** when the human rating reports the word as correctly produced, and a synonym was not used

4 Experiments and results

4.1 Dataset composition

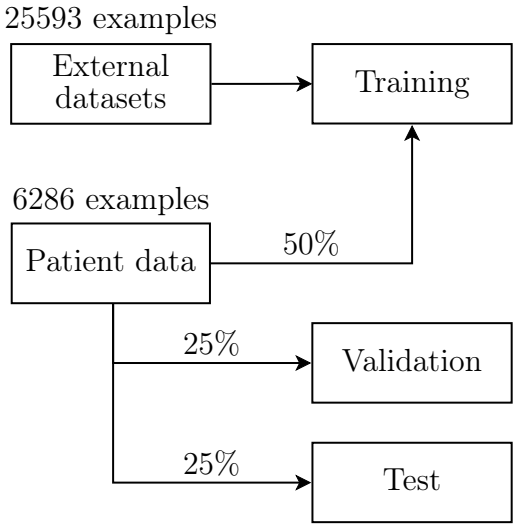


Figure 7: Training, validation and test datasets composition.

Our main dataset comprises naming test recordings from nine aphasic patients and four healthy controls, collected and evaluated as part of an ongoing study performed at the Brain State Decoding Lab in cooperation with the Neurology and Neurophysiology department of the University Medical Center Freiburg. The test is performed in German. Each test participant completed one or more separate test sessions, which consist of naming the 233 pictures presented to them in a fixed order, adding up to a total of 40 sessions and 9184 recordings (which isn't a multiple of 233 because several sessions were aborted prematurely). Each of these sessions was manually evaluated, resulting in each recording being associated with the correct response location (if present), phonological and semantic scores of the response, and the synonym if used. As the synonyms occur too rarely, we exclude them from the detection pipeline, focusing only on the 230 standard response words.

Due to the high quality of the manual labels, no particular data processing was necessary. The only cleaning step was the exclusion of excessively long word segments (longer than 2.3 sec). In total the dataset contains 6286 word examples, which is significantly less than the original recording count only because aphasic patients often struggle to name objects, leading to many missing responses. The main dataset totals to approximately 1.2 hours of speech.

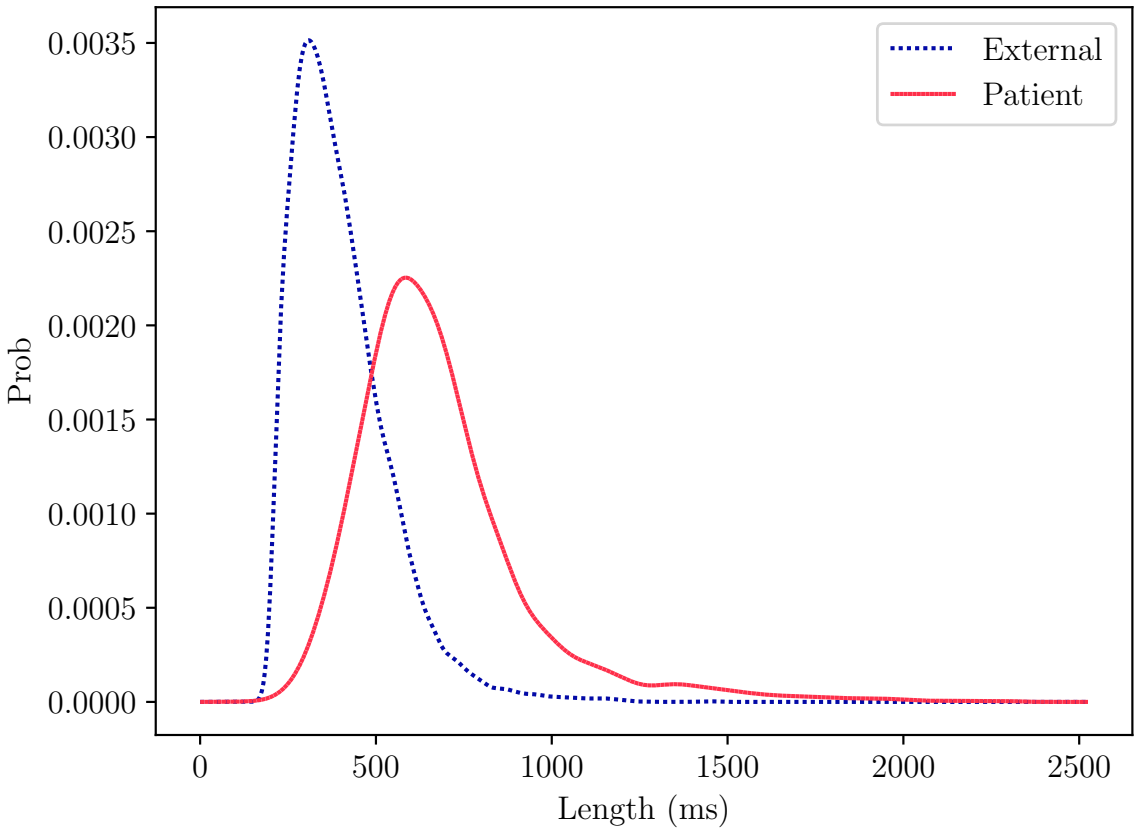


Figure 8: Word length distributions for external (healthy) and patient speech data, approximated using Gaussian kernel density estimation.

Since our main dataset contains a very low amount of data, we complement it with several datasets made available for academic users by the Bavarian Archive for Speech Signals repository[1], which operates as part of the CLARIN[2] – European Research Infrastructure for Language Resources and Technology project.

We include data from the following datasets: ALC [36], Phattsessionz [37], RVG1 [38], WaSeP [39], PhonDat1 and PhonDat2 [40], SprecherInnen [41], VM1 and VM2 (parts of the Verbmobil project [42]). We additionally use the German Spoken

[1] `https://clarin.phonetik.uni-muenchen.de/BASRepository/`
[2] `https://www.clarin.eu/`

Wikipedia Corpus (SWC) data [43]. These datasets contain a combination of spontaneous and read healthy speech, recorded in different, but mostly controlled environments and primarily with high quality equipment.

As several of the external datasets were not labeled manually, but time-aligned to textual transcriptions using a pre-trained speech recognition system (eg. as described in [43]), some of their labels can be inaccurate, which necessitates a more thorough data cleaning. An empirical evaluation showed that most of the word segments shorter than 200 ms for words with four or fewer letters and 250 ms for longer words were alignment errors, so those segments are excluded. The same was discovered for segments longer than 1.4 s, which are also excluded. In total 25593 external examples survived the cleaning, adding up to approximately 2.84 hours of speech data.

We can observe the difference in healthy and patient speech on Figure 8, judged by distributions of word pronunciation time approximated using Gaussian kernel density estimation. The significantly slower patient pronunciation speed can be explained by age differences, speech complications due to aphasia, and the different speech conditions – external data contains many instances of fluent speech, which is typically faster than the deliberate word production during naming.

The training/validation/test data split is depicted on Figure 7. All of the external data is used for training, as well as approximately 50% of the patient and healthy control data. The rest is split evenly into the validation and test sets. During splitting special care is taken not to divide one person's data between different datasets, to avoid overly optimistic generalization estimates. Overall we use the data of four patients and one control for the training set, two patients and two controls for the validation set, and three patients and one control for the test set.

4.2 Network structure and optimization

We take the best embedding network architecture reported by Settle and Livescu [20] as the starting point for our model investigation. The basic structure consists of two or more stacked RNN layers followed by one or more fully connected layers, the last of which outputs the acoustic embedding $f(X)$. As described in [20], the RNN hidden state after each input frame can already be considered a representation of the audio data seen so far, and the additional fully connected layers may serve to improve this representation. We use the final layer size to adjust the embedding dimensionality for Siamese networks, and for classifier networks it is fixed and equals

the number of distinct words in our detection vocabulary (230).

We investigate LSTM [44] and GRU [45] layers for the RNN part implementation. The fully connected layers use ReLU [46] activations, except for the final layer, which is linear for Siamese networks and log-softmax for classifier networks. All models were implemented using PyTorch [47] version 1.0rc1.

Optimization is performed with Adam [48], using a learning rate of 0.0015 for classifier networks and 0.001 for Siamese (determined experimentally by manual search between 0.01 and 0.0001, using validation set performance), the other Adam parameters were left in default state. The training progress is evaluated on the validation dataset after each epoch, which is used for learning rate scheduling: if the validation performance does not increase for six epochs (seven for the classifier networks), the learning rate is halved, until the minimum value of 0.00001.

For triplet-loss-based networks a training epoch consists of *#training examples* triplets, where for each word we randomly sample *average #examples per word* anchor examples and the same amount of positive examples, and positives are ensured to be different from anchors by re-randomizing the pairs with same examples. For each anchor-positive pair we randomly sample 10 negative examples, each from a different word, for the purpose of triplet mining as described in section 3.2.1, however only one triplet of these ten gets chosen for training. The triplets are presented in random order during training. We use a batch size of 960, where the first 80 examples in the batch are anchors, next 80 are positives, and the rest are negative examples. The margin parameter α is set to 0.5 for the cosine-distance-based triplet loss, and to 0.2 for the Euclidean distance variant (following [21] and [24] respectively).

For margin-loss-based networks a training epoch consists of *#training examples* $\times$ 4 positive pairs, and the same amount of negative pairs. We use a batch size of 870, and each batch is composed from examples of $870 \div 5 = 174$ different words, where we sample 5 random examples per word. For each example in the batch, each of the 4 other examples of the same word are used to create positive pairs, and 4 negative examples are chosen from all different-word examples in the batch according to the distance-based sampling described in section 3.2.1. The margin parameter α is set to 0.2, and the boundary parameter β is initialized with 1.2, while the cutoff λ is set to 0.5 (following [23]).

For classification networks a training epoch consists of *#training examples* samples, where for each word we sample *average #examples per word* examples with replacement, if the number of examples for this word is lower than average, and

without replacement otherwise. The examples are presented in random order, and the batch size is 768.

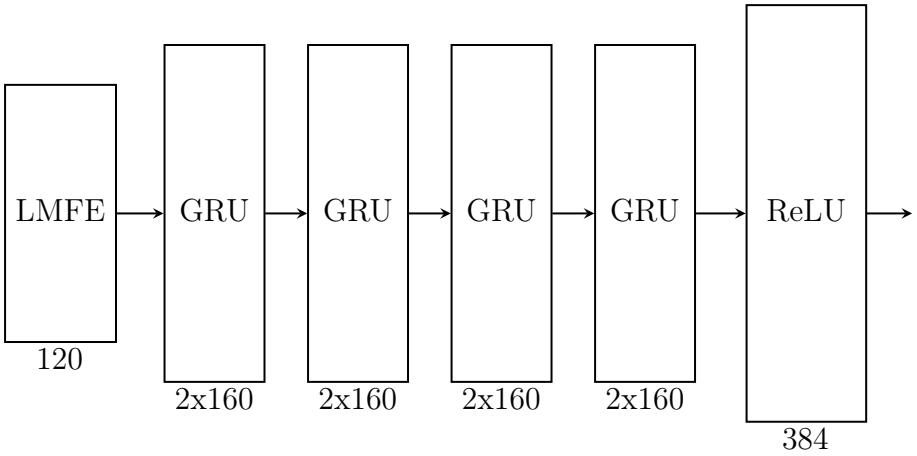


Figure 9: Best base network structure. Bottom numbers give layers sizes, and the GRU layers are bidirectional. Siamese networks append a 64-unit linear output layer to the base, and classifier networks use a 230-unit log-softmax layer.

We investigated different architectures by selecting best performing configurations of the classification network, as it trains significantly faster than the triplet-based Siamese networks (more than an order of magnitude difference), with the later experiments switching to the comparably fast margin loss training after it was implemented. Configurations with 2 to 5 stacked RNN layers and 1 to 4 fully connected layers (not counting the output layer) were tested on the validation set, with the following findings:

- Bidirectional RNN layers [49] lead to improved performance over unidirectional versions.
- Using dropout [50] between RNN layers improves performance and convergence speed, with the unit deactivation probability of 0.3 giving the best results for all tested configurations.
- Contrary to the results of [20], GRU lead to better performance than LSTM across all tested Siamese configurations.
- Again contrary to [20], using more than one fully connected layer degrades performance, with dropout between fully connected layers not helping to improve this situation.

The best found configuration is depicted on Figure 9, which is the configuration we use to report all of our results.

We optimized the embedding dimensionality for Siamese networks using the final network configuration by comparing between sizes from 2^4 to 2^9 and found 64-dimensional embeddings to perform the best when coupled with the margin loss, while for the other losses the dimensionality choice did not seem to have a large influence on performance.

4.3 Embedding learning

We first investigate the embedding learning approaches introduced in Section 3.2 and compare the cosine and Euclidean distance variants of the triplet loss, as well as the margin loss. Figure 10 shows plots of the network training progress when using the aforementioned losses.

It is important to note here that the cosine-distance-based Siamese networks fail to learn when starting from random initializations, as the models collapsed to all input segments having the same representations in all training attempts. Following [20], we use the weights of the best previously trained classifier network with the same architecture for their initialization (the output layer is discarded, otherwise the weights are reused as is).

The Euclidean distance triplet loss variant overcomes this limitation, however its performance appears to be worse than of the previous approach. This can be explained by a lack of pretraining, as with a similar classifier-based initialization this

	Val AP	Test AP	Val accuracy	Test accuracy
Triplet cosine	0.494 ± 0.016	0.491 ± 0.022	0.690 ± 0.011	0.687 ± 0.010
Triplet Euclidean	0.438 ± 0.014	0.412 ± 0.007	0.672 ± 0.006	0.663 ± 0.019
Margin loss (ND)	0.481 ± 0.009	0.456 ± 0.006	0.691 ± 0.003	0.687 ± 0.011
Margin loss	0.525 ± 0.009	0.514 ± 0.009	0.726 ± 0.006	0.724 ± 0.008
Classifier	0.309 ± 0.009	0.287 ± 0.002	0.650 ± 0.007	0.662 ± 0.014

Table 1: Performance comparison for Siamese and classifier networks. Mean±std for the epoch with the best validation performance is shown, where the statistics are calculated over three independent runs for each configuration. ND stands for "no dropout".

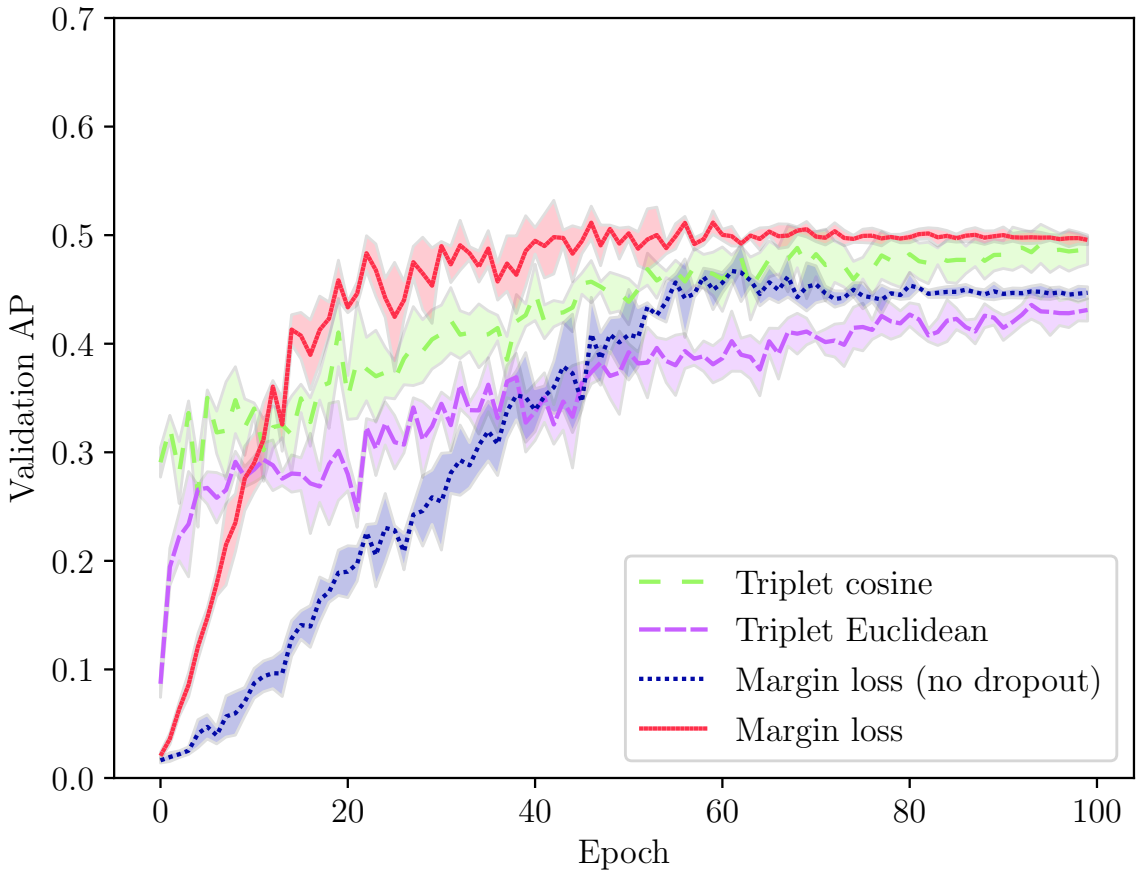


Figure 10: Siamese network learning curves. The mean is shown in bold, the shaded regions give ±1 standard deviation. The mean and standard deviation for each configuration are calculated over 3 independent runs.

loss achieves comparable levels of performance.

The adaptive margin loss surpasses all the other approaches, both in convergence speed and stability as well as performance. The distance-based pair sampling used with it leads to a massive training speedup (∼25 s per epoch instead of ∼300 s for the triplet loss), as resource-intensive triplet mining is no longer required. We add a configuration without dropout between recurrent layers to the comparison to demonstrate that its omission leads to much slower learning progress and worse final performance.

A comparison of validation and test set performances for all considered configurations is shown in Table 1, where for each run we take the epoch with the best validation set performance as the final model. As we can see, all Siamese networks outperform the classifier network, both in average precision and in classification accuracy, and the margin loss is the clear winner in all categories. While it is not

unexpected that the classifier-based embeddings possess worse discriminative quality, as their training does not directly optimize for it, the worse classification performance may appear surprising. However, we can attribute this to the more sample effective training of Siamese networks, as even a single word example can lead to as many training pairs as there are other samples in the dataset, amplifying the effective dataset size.

4.3.1 Noise augmentation

Adding random noise at neural network input can be considered a form of dataset augmentation, as described by Goodfellow et al. [51]. Considering the low amount of data in our training dataset, we explore a simple augmentation approach based on adding noise to the extracted speech features.

Our approach consists of adding Gaussian-distributed noise with zero mean and variance equal to feature variance computed over all frames in the training dataset. A noise vector is sampled for each example independently and added to all its frames, i.e. every frame of a specific audio segment is perturbed by the same noise. We investigated noise magnitudes between 0.1 to 1.5, applied by multiplying the sampled noise vectors with a constant factor chosen from this range before training. We also explore the effect of changing the probability of applying these perturbations by comparing the results of training runs in which 30%, 50%, 70%, or 100% of the batches were modified with noise.

We explored the aforementioned parameter ranges on margin-loss-based Siamese networks and found the most improvement for noise magnitude of 1.0 applied in 50% of the batches. Table 2 shows a detailed performance comparison for all considered Siamese losses when using the best found noise parameters, from which we can see that the margin-loss-based network and the classifier network benefit from noise augmentation the most. As the triplet-loss-based networks perform triplet mining by selecting the highest loss negative example, we presume that the noisiest examples always get selected for training, which may explain the lack of performance improvement for these networks.

Considering the improvement of margin loss due to noise augmentation, in conjunction with its much higher computational effectiveness, we choose margin loss as the best Siamese loss variant and continue further experiments with it alone.

	Val AP	Test AP	Val accuracy	Test accuracy
Triplet cosine	0.504 ± 0.008	0.482 ± 0.005	0.712 ± 0.006	0.694 ± 0.004
Triplet Euclidean	0.430 ± 0.006	0.393 ± 0.014	0.664 ± 0.010	0.645 ± 0.012
Margin loss	0.585 ± 0.014	0.543 ± 0.009	0.750 ± 0.011	0.734 ± 0.008
Classifier	0.362 ± 0.008	0.305 ± 0.003	0.693 ± 0.002	0.670 ± 0.003

Table 2: Performance comparison for Siamese and classifier networks with noise augmentation.

4.3.2 Variational dropout

As was demonstrated on Figure 10, using dropout between the recurrent layers improves the learning dynamics and the final performance of our best network configuration. The standard approach samples a new dropout mask for each timestep, which heavily disrupts the information flow if used between recurrent connections, which is why it is applied only at the inputs and outputs [52]. While this works in practice, the usage of this dropout technique is poorly grounded for recurrent networks, as argued by Gal and Ghahramani [53], who proposed a dropout approach specifically derived for RNNs.

Their proposed technique, to which we will refer to as "variational dropout", drops the same network units at each timestep, i.e. the dropout mask is sampled once per input sequence, instead of once per each input timestep. The consistent masks should conserve the information flow over long sequences and enabled Gal and Ghahramani to apply dropout between recurrent connections. We implemented the tied-weights variant of their approach (using the same mask for different appearances of input and hidden state vectors in the GRU equations, instead of sampling different dropout masks for different gates) and evaluated it on the best model determined in the previous experiments. While using this variant of dropout on the GRU hidden states lead to significantly worse performance, using it in the same manner as conventional dropout did deliver a small performance increase, as shown in Table 3. The fact that the performance difference is more evident on the test set seems to indicate that variational dropout is more effective at reducing overfitting. We adopt this technique for our further experiments.

	Val AP	Test AP	Val accuracy	Test accuracy
Margin	0.525 ± 0.009	0.514 ± 0.009	0.725 ± 0.007	0.724 ± 0.007
Margin+VD	0.547 ± 0.007	0.540 ± 0.010	0.727 ± 0.006	0.730 ± 0.007
Margin+NA	0.585 ± 0.014	0.543 ± 0.009	0.750 ± 0.011	0.735 ± 0.008
Margin+VD+NA	0.595 ± 0.016	0.574 ± 0.009	0.754 ± 0.006	0.749 ± 0.005
Classifier	0.309 ± 0.009	0.287 ± 0.002	0.650 ± 0.007	0.662 ± 0.014
Classifier+VD	0.342 ± 0.002	0.313 ± 0.013	0.673 ± 0.003	0.685 ± 0.010
Classifier+NA	0.362 ± 0.008	0.305 ± 0.003	0.693 ± 0.002	0.670 ± 0.003
Classifier+VD+NA	0.380 ± 0.010	0.324 ± 0.014	0.723 ± 0.007	0.699 ± 0.013

Table 3: Performance comparison for margin-loss-based Siamese and classifier networks with variational dropout and noise. VD stands for "variational dropout", NA stands for "noise augmentation".

	Val AP	Test AP	Val accuracy	Test accuracy
Margin	0.547 ± 0.007	0.540 ± 0.010	0.726 ± 0.007	0.731 ± 0.008
Margin+PA	0.568 ± 0.014	0.583 ± 0.008	0.731 ± 0.007	0.745 ± 0.002
Margin+NA	0.595 ± 0.016	0.574 ± 0.009	0.753 ± 0.006	0.747 ± 0.005
Margin+PA+NA	0.598 ± 0.003	0.591 ± 0.008	0.747 ± 0.003	0.750 ± 0.004

Table 4: Performance comparison for the best performing Siamese configuration with parts augmentation. All the listed configurations use variational dropout, omitted due to space considerations. PA stands for "parts augmentation".

4.3.3 Parts augmentation

We demonstrate in section 4.1 that a major mismatch between healthy and patient speech data lies in the speaking rate difference, with the patients on average taking much longer to pronounce words. This may present difficulties for recognition, as our models need to learn to become invariant to pronunciation speed using the limited and mostly quickly-pronounced examples available. We consider an online dataset augmentation technique intended to promote this invariance, which we will refer to as "parts augmentation".

The proposed augmentation consists in performing triplet- or pair-based training on proportional parts of the words in some of the training batches, instead of using the whole example data. We assume that the speaking rate is constant within a word, so that, for example, 50% of one spoken word instance roughly correspond phonetically to the 50% of another spoken word instance, even if the actual word durations differ significantly. The assumption is then that by enforcing the similarity relations introduced in subsection 3.2.1 between proportional parts of example pairs (or triplets) we could encourage the desired speaking rate invariance. We implement this idea by taking the first 20%, 30%,...,90% of the example frames (randomly selected for a batch) in 50% of the training batches.

We use the previously determined best performing combination of margin loss and variational dropout and compare the effect of parts augmentation with and without noise, with the results shown in Table 4. As we can see, the proposed augmentation shows a small improvement over the base configuration, while combining it with noise leads to diminishing results. Overall, the performance differences are too small to make a decisive conclusion, but we still use the combination of all previous augmentations with the proposed one for the final Siamese configuration, which will be used for response detection.

4.3.4 Clean dataset comparison

While the previous experiments show performance levels similar to the results of Settle and Livescu [20] who achieve a best AP of 0.671, our results are not directly comparable due to two factors: 1) the training and evaluation are done with atypical (aphasic) speech, 2) our patient data is very noisy due to suboptimal recording setup and equipment, unlike the typical data available in academic speech datasets. Therefore, we would like to benchmark our approach on non-aphasic high quality

	Val AP	Test AP	Val accuracy	Test accuracy
Margin	0.878 ± 0.002	0.863 ± 0.002	0.891 ± 0.002	0.897 ± 0.003
Margin+VD	0.883 ± 0.004	0.869 ± 0.006	0.885 ± 0.007	0.894 ± 0.005
Margin+NA	0.882 ± 0.002	0.861 ± 0.001	0.889 ± 0.003	0.891 ± 0.002
Margin+VD+NA	0.892 ± 0.000	0.868 ± 0.003	0.891 ± 0.004	0.895 ± 0.001
Classifier	0.655 ± 0.011	0.595 ± 0.011	0.811 ± 0.002	0.807 ± 0.004
Classifier+VD	0.656 ± 0.005	0.596 ± 0.005	0.819 ± 0.002	0.812 ± 0.002
Classifier+NA	0.651 ± 0.013	0.600 ± 0.014	0.823 ± 0.001	0.812 ± 0.006
Classifier+VD+NA	0.661 ± 0.003	0.606 ± 0.007	0.826 ± 0.004	0.819 ± 0.004

Table 5: Performance comparison for margin-loss-based Siamese and classifier networks on the clean dataset.

speech data to explore its performance potential.

As we do not have access to the same dataset used in [20], we construct a surrogate dataset from the external data gathered to supplement the patient recordings, which we intend to be very similar to the training dataset we used in the previous experiments. We use all of the external datasets, except the largest – SWC (which is used for validation and test data), to select a new training dataset as follows: new word classes are selected randomly from words vaguely matching the naming test vocabulary: same or plus one number of characters, also nouns, with at least 1.5 times as many examples as the corresponding original word (to account for cleaning removing a large portion of the potential examples) and with at least 1.5 times as many examples in SWC as in the original validation set. After the words are chosen, we collect their audio data and perform the same cleaning as described in section 4.1. From the surviving data we take up to as many examples for each word to the new training set as was available in the original training set for the corresponding word, and up to 35 examples to the test set. The validation set is composed from the remaining SWC examples of the new words, also with an upper limit of 35 examples per word. The final dataset sizes are 25788 examples in the training set, 4311 examples in the validation set, and 6725 examples in the test set.

A performance comparison for our best model with and without augmentations is shown in Table 5, where parts augmentation is omitted as we assume the speaking rate difference to be absent in this data. We can see that the previously useful augmentations do not make a large difference in performance on the clean dataset,

which may be explained by the fact that the issues they were helping to mitigate are not present in this data.

Overall the performance level is much higher and is close to the more recent results of He et al. [54], however a proper comparison would require training and evaluation on the same dataset as used in their study.

4.4 Response detection

We use the acoustic embedding network with the best validation set performance for the response detection task, which is the margin-loss-based Siamese model trained employing all the previously described augmentations. The validation performance values for the chosen model are 0.617 AP and a classification accuracy of 0.759 (75.9% correctly classified word segments).

We found that the used voice activity detection approach heavily suffers from the high noise levels in the patient recordings, leading to the detected speech regions often being padded by noise from one or both ends, and difficulties in separating individual speech regions due to the noise between them. We attempt to correct this by running VAD on copies of the recordings denoised using *sox*[3], an open-source command line utility for sound processing. Denoising is done based on a noise profile extracted from a pure noise region manually selected from a validation set recording.

Testing the sliding window segmentation approach on the validation set recordings showed that the detected segments tend to contain long regions silence or noise after the keyword audio, which is ignored by the embedding network and does not lead to the silence-containing segments having a worse distance score. We attempt to improve the detected segments by removing these regions using voice activity detection.

4.4.1 Segmentation methods evaluation

We evaluate both input recording segmentation approaches on the test set recordings, using the thresholding types introduced in subsection 3.5.2. The VAD-based method's performance is shown in Table 6, and the sliding window method's in Table 7. As we can see, the sliding window approach delivers the superior performance over all detection thresholds, and the beta-based thresholding gives the best tradeoff

[3] `http://sox.sourceforge.net`

Thr_{imm}	Thr_{exh}	TP	FP	TN	FN	Prec	Rec	F1	Acc
0.10	0.20	90	51	856	1475	0.64	0.06	0.11	38.27
0.20	0.30	223	187	813	1249	0.54	0.15	0.24	41.91
0.30	0.40	331	312	763	1066	0.51	0.24	0.32	44.26
0.40	0.50	418	436	711	907	0.49	0.32	0.38	45.67
0.40	0.60	487	609	629	747	0.44	0.39	0.42	45.15
0.40	0.70	557	872	506	537	0.39	0.51	0.44	43.00
0.40	0.80	584	1287	290	311	0.31	0.65	0.42	35.36
0.40	0.90	600	1686	88	98	0.26	0.86	0.40	27.83
0.40	1.00	607	1842	10	13	0.25	0.98	0.40	24.96
$\beta - .2$	$\beta + .2$	559	887	504	522	0.39	0.52	0.44	43.00

Table 6: Response detection performance using VAD-based segmentation. True and false positive and negative counts are shown, together with detection precision, recall, F1 score and total accuracy in percent.

between precision and recall. The final threshold selection would depend on the rater's choice as to whether a lower false positive or a lower false negative detection rate is preferred.

We discovered that finding the exact word location is challenging for our system: relaxing the boundary closeness condition to 300 ms shows that a large proportion of the responses are detected correctly, but with low-quality word start or end position estimates. The performance in relaxed conditions is shown in Table 8.

4.4.2 Per-patient adaptation test

As mentioned in subsection 3.1.3, one argument in favor of template-based methods is their adaptability to new speakers and environmental conditions, realized by incorporating reference examples from new data. Our dataset contains several separate naming test sessions for each patient, which enables us to test the adaptability of our system in a straightforward manner: we perform a cross-validation test, in which the data from one of the patient's sessions is incorporated into the reference example pool, and we calculate the response detection performance on all other sessions from this patient. We cycle the session used for new reference data and average the obtained results, which are then compared to the average performance over the same folds when no new reference examples were used. Beta-based thresholding

Thr_{imm}	Thr_{exh}	TP	FP	TN	FN	Prec	Rec	F1	Acc
0.10	0.20	207	113	829	1323	0.65	0.14	0.22	41.91
0.20	0.30	495	366	722	889	0.57	0.36	0.44	49.23
0.30	0.40	651	562	629	630	0.54	0.51	0.52	51.78
0.40	0.50	772	753	542	405	0.51	0.66	0.57	53.16
0.40	0.60	842	914	449	267	0.48	0.76	0.59	52.22
0.40	0.70	889	1120	316	147	0.44	0.86	0.58	48.75
0.40	0.80	915	1366	142	49	0.40	0.95	0.56	42.76
0.40	0.90	925	1515	27	5	0.38	0.99	0.55	38.51
0.40	1.00	926	1542	4	0	0.38	1.00	0.55	37.62
$\beta - .2$	$\beta + .2$	890	1127	316	139	0.44	0.86	0.58	48.79

Table 7: Response detection performance using sliding window segmentation. True and false positive and negative counts are shown, together with detection precision, recall, F1 score and total accuracy in percent.

Thr_{imm}	Thr_{exh}	TP	FP	TN	FN	Prec	Rec	F1	Acc
0.10	0.20	237	83	829	1323	0.74	0.15	0.25	43.12
0.20	0.30	610	251	722	889	0.71	0.41	0.52	53.88
0.30	0.40	822	391	629	630	0.68	0.57	0.62	58.70
0.40	0.50	1003	522	542	405	0.66	0.71	0.68	62.50
0.40	0.60	1104	652	449	267	0.63	0.81	0.71	62.82
0.40	0.70	1181	828	316	147	0.59	0.89	0.71	60.56
0.40	0.80	1225	1056	142	49	0.54	0.96	0.69	55.30
0.40	0.90	1241	1199	27	5	0.51	1.00	0.67	51.29
0.40	1.00	1243	1225	4	0	0.50	1.00	0.67	50.44
$\beta - .2$	$\beta + .2$	1184	833	316	139	0.59	0.89	0.71	60.68

Table 8: Response detection performance using sliding window segmentation with closeness condition relaxed to 300 ms. True and false positive and negative counts are shown, together with detection precision, recall, F1 score and total accuracy in percent.

is used here to obtain a single performance figure and simplify the comparison.

We compute the performance values by adding the true and false positives and negatives from each fold and calculating the precision, recall, F1 score, and accuracy from the total values, as suggested by Forman and Scholz in [55] to obtain unbiased averages.

We compare two ways to incorporate the new reference data: 1) taking *only* the new examples when available and falling back to the training set examples if the word was not produced in the new reference session, 2) adding the new example to the training set reference pool with a 50% weight, i.e. during average distance calculations the new reference contributes as much as all the old references.

We perform this experiment on two patients from the validation dataset, both of whom completed four naming test sessions. The results are shown in Table 9. As we can see, using only the new examples somewhat degrades the performance, however the averaging adaptation approach leads to a consistent improvement. Although this improvement is modest, we used only one new session here, and with additional data we can hope for a stronger effect. Furthermore, we do not optimize the adaptation process here, and a different weighting of the new examples may lead to better performance.

	Type	Precision	Recall	F1	Accuracy
Patient A	Non-adapted	0.597	0.810	0.687	61.803
	Only new	0.543	0.815	0.652	57.690
	Average with new	0.614	0.809	0.698	62.732
Patient B	Non-adapted	0.493	0.737	0.591	57.035
	Only new	0.462	0.778	0.580	55.460
	Average with new	0.501	0.761	0.604	58.110

Table 9: Per-patient adaptation test results. Accuracy shown in percent.

The main advantage of this adaptation is its simplicity, as the new data can be integrated without retraining and with minimal supervision, which is desirable for a practical system.

5 Conclusion

In this work we have considered several speech recognition approaches in the context of response detection and localization in naming tests for aphasic speech patients and advocated a template matching solution based on acoustic embeddings as a flexible and extendable approach for response detection. We compared several embedding-learning methods previously used in speech recognition and computer vision tasks and selected margin-loss-based Siamese networks as the best-performing embedding extraction approach. Several techniques were considered to improve the learned embedding quality and support generalization between healthy and aphasic speech.

Two variants of the response detection system built using the learned embeddings were considered, with the sliding window approach beating VAD-based segmentation in both performance and speed. While the final recognition rates are low compared to human-level performance on this task, the detection system can be useful by providing suggestions for possible response location, since a large proportion of all responses could be at least approximately located, as shown in subsection 4.4.1. The developed system is compatible with an existing application for manual naming test rating and can be used to generate response detection guesses viewable in the application.

We benchmarked our acoustic embedding learning approach on a dataset composed of high quality healthy speech recordings and established that it achieves performance levels competitive with recent state-of-the-art results in acoustic embedding learning.

6 Future work

In this chapter we outline several possible future investigation directions for response detection and quality assessment.

6.1 Response detection

Since our current solution does not account for synonym usage, one possible improvement is to enlarge the recognition vocabulary to support them. The main obstacle for synonym inclusion in this work was their rarity, as patients often use dialectic terms not encountered in the external datasets we used to collect additional examples for training. However, analysis of response detection accuracy on the test set shows that very low training example count does not majorly impair performance, as demonstrated in Table 10, so external data collection may not be mandatory for vocabulary extension.

Table 10 suggests another improvement idea: performance maximum is observed for words with 25–50 examples, for which there is a balanced amount of external and patient samples, and for words with a massive amount of examples the performance inexplicably drops. We speculate that when external examples dominate the reference count, the distance comparisons in the response detection stage start favoring typical speech too strongly, lowering performance on aphasic speech data. A simple solution would be to give equal weight to the external and patient examples in distance averaging, however further investigation is needed.

A major problem for recognition accuracy seems to be the high levels of noise in the patient data, as evident from the clean dataset comparison. Noise-resistant acoustic features have been strongly sought for in speech recognition community, with promising developments in this area. Preliminary experiments with power-normalized cepstral coefficients (PNCC) [56] designed for robust speech recognition failed to show an improvement, or even match the performance of our current feature extraction method, however a likely reason for this may be undiscovered bugs in our PNCC

Training example count	Accuracy
3–5 examples	50.
5–10 examples	42.405
10–25 examples	49.089
25–50 examples	54.902
50–100 examples	54.31
$>$ 100 examples	43.202

Table 10: Word detection accuracy on the test set recordings per training example count. Accuracy shown in percent.

implementation.

Another possible research direction may be the introduction of an explicit background audio model to our embedding extraction system. The current approach seeks to maximize discriminability between different word classes, however external words and non-word sounds are completely omitted from the training pipeline. Following the keyword-filler idea used for HMM-based keyword search, as mentioned in subsection 3.1.1, may be beneficial for an embedding-based approach as well.

6.2 Speech quality assessment

We can divide the ideas for next steps towards speech quality assessment in two categories: 1) classification-based rating and 2) multi-task learning approaches. All of these approaches would need to take into account that the phonological quality rating includes information from the pronunciation context, eg. the presence of restarts and self-corrections, so the training data would need to be adapted to contain this kind of information.

6.2.1 Classification-based rating

A simple approach could consist of extending the kNN-based classifier we used for between-word classification to work with quality rating classes instead. As each word example in the patient data is associated with an integer rating between 1 to 4 (zero-rated examples are typically not associated with an audio segment, so we do not consider them here), we could attempt to directly classify new pronunciations based on closeness to the training set examples in hopes that similarity-based training

captures factors relevant to quality assessment. Another possibility is to fine-tune the learned embeddings with a classifier network trained on quality labels to enrich them with quality-relevant information.

Considering that most of the examples receive very high ratings, as shown in Table 11, it could be hard to obtain a meaningful classification between the given four quality classes. Instead, a possible solution is to divide the detected responses into two classes: 1) high quality, no manual review required, 2) low quality, detailed inspection necessary. An approach like this could save a large portion of the rater's time, as most of the examples would no longer require their attention.

Rating	Example count
1	1
2	29
3	572
4	4534

Table 11: Quality rating statistics over the training and validation sets.

6.2.2 Multi-task learning

Several recent works explore learning acoustic embeddings in a multi-task learning setting to enrich the learned embeddings with character-level word similarity information [54] or with phonetic information [57], allowing to judge the corresponding similarities based on between-embedding distances. Considering that the phonological rating is mostly based on the fraction of correctly pronounced phonemes, phoneme-level closeness should be directly usable for rating purposes. However, we could also perform similar multi-task learning using quality ratings, forcing the learned distance to respect different gradations of pronunciation quality.

Bibliography

[1] H. Goodglass, E. Kaplan, and B. Barresi, *The Assessment of Aphasia and Related Disorders*. The Assessment of Aphasia and Related Disorders, Lippincott Williams & Wilkins, 2001.

[2] A. Abad, A. Pompili, A. Costa, I. Trancoso, J. Fonseca, G. Leal, L. Farrajota, and I. P. Martins, "Automatic word naming recognition for an on-line aphasia treatment system," *Computer Speech & Language*, vol. 27, no. 6, pp. 1235–1248, 2013.

[3] D. Le, K. Licata, E. Mercado, C. Persad, and E. M. Provost, "Automatic analysis of speech quality for aphasia treatment.," in *ICASSP*, pp. 4853–4857, Citeseer, 2014.

[4] D. Le, K. Licata, C. Persad, E. M. Provost, D. Le, K. Licata, C. Persad, and E. M. Provost, "Automatic assessment of speech intelligibility for individuals with aphasia," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 24, pp. 2187–2199, Nov. 2016.

[5] K. Fraser, F. Rudzicz, N. Graham, and E. Rochon, "Automatic speech recognition in the diagnosis of primary progressive aphasia," in *Proceedings of the fourth workshop on speech and language processing for assistive technologies*, pp. 47–54, 2013.

[6] D. Jurafsky and J. H. Martin, *Speech and Language Processing (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2009.

[7] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *Ann. Math. Statist.*, vol. 41, pp. 164–171, 02 1970.

[8] X. L. Aubert, "An overview of decoding techniques for large vocabulary continuous speech recognition," *Comput. Speech Lang.*, vol. 16, pp. 89–114, Jan. 2002.

[9] D. A. James and S. J. Young, "A fast lattice-based approach to vocabulary independent wordspotting," in *ICASSP (1)*, pp. 377–380, IEEE Computer Society, 1994.

[10] D. R. H. Miller, M. Kleber, C.-L. Kao, O. Kimball, T. Colthurst, S. A. Lowe, R. M. Schwartz, and H. Gish, "Rapid and accurate spoken term detection.," in *INTERSPEECH*, 2007.

[11] J. Rohlicek, W. Russell, S. Roukos, and H. Gish, "Continuous hidden markov modeling for speaker-independent word spotting," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, pp. 627 – 630, IEEE, 1990.

[12] R. C. Rose and D. B. Paul, "A hidden markov model based keyword recognition system," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 1, pp. 129–132, 01 1990.

[13] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *in Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4087–4091, IEEE, 2014.

[14] J. Bridle, "An efficient elastic-template method for detecting given words in running speech," *Brit. Acoust. Soc. Meeting*, pp. 1–4, 01 1973.

[15] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, pp. 43–49, February 1978.

[16] K. Levin, K. Henry, A. Jansen, and K. Livescu, "Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings," in *Workshop on Automatic Speech Recognition and Understanding, ASRU 2013 - Proceedings*, pp. 410–415, IEEE, 12 2013.

[17] A. L. Maas, S. D. Miller, T. M. O'Neil, A. Y. Ng, and P. Nguyen, "Word-level acoustic modeling with convolutional vector regression," in *International*

Conference on Machine Learning (ICML), Representation Learning Workshop, 2012.

[18] S. Bengio and G. Heigold, "Word embeddings for speech recognition," in *Proceedings of the 15th Conference of the International Speech Communication Association, Interspeech*, 2014.

[19] H. Kamper, W. Wang, and K. Livescu, "Deep convolutional acoustic word embeddings using word-pair side information," *CoRR*, vol. abs/1510.01032, 2015.

[20] S. Settle and K. Livescu, "Discriminative acoustic word embeddings: Recurrent neural network-based approaches," *CoRR*, vol. abs/1611.02550, 2016.

[21] S. Settle, K. Levin, H. Kamper, and K. Livescu, "Query-by-example search with discriminative neural acoustic word embeddings," *CoRR*, vol. abs/1706.03818, 2017.

[22] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, "Signature verification using a "Siamese" time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 04, pp. 669–688, 1993.

[23] C. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl, "Sampling matters in deep embedding learning," *CoRR*, vol. abs/1706.07567, 2017.

[24] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *CoRR*, vol. abs/1503.03832, 2015.

[25] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 1735–1742, June 2006.

[26] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *CoRR*, vol. abs/1703.07737, 2017.

[27] "The sphere game in n dimensions." `http://faculty.madisoncollege.edu/alehnen/sphere/hypers.htm`. Accessed: 24.01.2019.

[28] K. Levin, A. Jansen, and B. V. Durme, "Segmental acoustic indexing for zero resource keyword search," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5828–5832, April 2015.

[29] "WebRTC project." `https://webrtc.org/`.

[30] "Python interface to the WebRTC Voice Activity Detector." `https://github.com/wiseman/py-webrtcvad`.

[31] L. R. Rabiner and R. W. Schafer, *Introduction to Digital Speech Processing*. Hanover, MA, USA: Now Publishers Inc., 2007.

[32] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *ACOUSTICS, SPEECH AND SIGNAL PROCESSING, IEEE TRANSACTIONS ON*, pp. 357–366, Aug. 1980.

[33] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[35] N. Zeghidour, N. Usunier, G. Synnaeve, R. Collobert, and E. Dupoux, "End-to-end speech recognition from the raw waveform," *CoRR*, vol. abs/1806.07098, 2018.

[36] F. Schiel, C. Heinrich, and S. Barfüsser, "Alcohol Language Corpus: The First Public Corpus of Alcoholized German Speech," *Lang. Resour. Eval.*, vol. 46, pp. 503–521, Sept. 2012.

[37] C. Draxler and A. Steffen, "Phattsessionz: recording 1000 adolescent speakers in schools in Germany," 01 2005.

[38] S. Burger and F. Schiel, "RVG 1 – A Database for Regional Variants of Contemporary German," in *PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON LANGUAGE RESOURCES AND EVALUATION*, pp. 1083–1087, 1998.

[39] B. Wendt and H. Scheich, "The "Magdeburger Prosodie Korpus" - a spoken language corpus for fMRI-studies," in *Speech Prosody 2002, International Conference*, pp. 699–701, 01 2002.

[40] W. J. Hess, K. J. Kohler, and H.-G. Tillmann, "The Phondat-verbmobil speech corpus.," in *Proceedings of the Eurospeech 1995*, 01 1995.

[41] "Spoken production of gender-neutral nouns in German." `http://hdl.handle.net/11022/1009-0000-0003-FF39-F`.

[42] S. Burger, K. Weilhammer, F. Schiel, and H. G. Tillmann, "Verbmobil data collection and annotation," in *Verbmobil: Foundations of speech-to-speech translation*, pp. 537–549, Springer, 2000.

[43] A. Köhn, F. Stegen, and T. Baumann, "Mining the Spoken Wikipedia for Speech Data and Beyond," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)* (N. C. C. Chair), K. Choukri, T. Declerck, M. Grobelnik, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis, eds.), (Paris, France), European Language Resources Association (ELRA), May 2016.

[44] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[45] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014.

[46] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[47] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[48] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[49] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *Trans. Sig. Proc.*, vol. 45, pp. 2673–2681, Nov. 1997.

[50] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[51] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[52] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *CoRR*, vol. abs/1409.2329, 2014.

[53] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in neural information processing systems*, pp. 1019–1027, 2016.

[54] W. He, W. Wang, and K. Livescu, "Multi-view recurrent neural acoustic word embeddings," *CoRR*, vol. abs/1611.04496, 2016.

[55] G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 1, pp. 49–57, 2010.

[56] C. Kim and R. M. Stern, "Power-Normalized Cepstral Coefficients (PNCC) for Robust Speech Recognition," *IEEE/ACM TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING,*, vol. 24, pp. 1315–1329, 2016.

[57] H. Lim, Y. Kim, Y. Jung, M. Jung, and H. Kim, "Learning acoustic word embeddings with phonetically associated triplet network," *arXiv preprint arXiv:1811.02736*, 2018.