Karlsruher Institut für Technologie

Audio Segmentation for Robust Real-Time Speech Recognition Based on Neural Networks

Bachelor's Thesis of

Micha Wetzel

at the Department of Informatics
Institute for Anthropomatics and Robotics, Interactive Systems Labs

Reviewer:	Prof. Dr. Alexander Waibel
Second reviewer:	Prof. Dr.-Ing. Rainer Stiefelhagen
Advisor:	M.Sc. Matthias Sperber

15. June 2016

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 15/6/2016

. . . .

(Micha Wetzel)

Abstract

Multimedia content hurts the recognition accuracy and speed of automatic speech recognition (ASR) systems. This bachelor thesis introduces a segmenter to increase the performance of an real-rime ASR system by detecting music and noise segments in an audio source and replacing it with silence. A two step approach is proposed, consisting of frame classification and smoothing. Audio frames of size 10 milliseconds are classified as speech, music or noise with a classification model. Multiple settings with neural nets and support vector machines as model are compared, resulting in an classification accuracy of 87%. In the second step the smoothing algorithm considers the temporal context to prevent rapid class fluctuations. The proposed segmenter yields a transcript quality of an ASR system en-par with manual removal of the music segments, while maintaining a real-time applicable delay of 270 milliseconds.

Zusammenfassung

Automatische Spracherkenner (ASR) reagieren generell schlecht auf eine Audioeingabe, die Musik oder Störgeräusche enthält. Die Spracherkennung verzögert sich dabei, da es keine wahrscheinlichen Thesen gibt. Zusätzlich nimmt dadurch die Anzahl der falsch erkannten Wörtern zu. Es ist daher erstrebenswert Abschnitte herauszufiltern, die Musik oder Störgeräusche enthalten, wodurch die Wortfehlerrate (WER) ebenso wie die Verzögerung verringert wird. Besonders wichtig ist dies im Kontext der Echtzeit-Spracherkennung, bei der das Transkript mit wenigen Sekunden Verzögerung verfügbar sein muss, um von Nutzen zu sein. Eine Verzögerung von maximal 5 Sekunden wird im Allgemeinen noch als akzeptabel bewertet.

In dieser Arbeit wird ein Segmentierungsverfahren zur Unterscheidung von Sprache, Musik und Störgeräuschen in einem Audiostream präsentiert. Dieses Verfahren ist dazu geeignet als Vorverarbeitungsschritt für einen Echtzeit-Spracherkenner genutzt zu werden. Das Segmentierungsverfahren nutzt einen zweistufigen Ansatz: Zuerst wird der Audiostream in Frames aufgeteilt und diese klassifiziert, danach werden die Klassifizierungen geglättet. Zur Klassifizierung werden markante Merkmale aus den Audioframes extrahiert, welche auf Mel Frequency Cepstral Coefficients und der Zero-crossing Rate basieren. Zwei Klassifikationsmodelle werden evaluiert, zum einen Neuronale Netze, zum anderen Support Vector Machines. Dabei werden verschiedene Parameter der Merkmalsextraktion sowie der Klassifikationsmodelle optimiert, wobei der Augenmerk auf der Fehlerfreiheit und einer geringen Latenz liegt. Da der Spracherkenner selbst im Allgemeinen schon einige Sekunden Latenz erzeugt, muss die durch den Segmentierungsalgorithmus entstehende Latenz sehr klein gehalten werden (idealerweise unter 0.5 Sekunden). Im zweiten Schritt wird die Klassifizierung der Frames geglättet, um Segmente von Audioklassen zu erhalten und kleinere Falschklassifikationen auszumerzen. Dafür werden die Frames kurz vor und nach dem aktuellen Frame zur Klassifikation zu Rate gezogen. Zusätzlich werden Frames der letzten Sekunden mit in Betracht gezogen. Dieser Glättungsschritt ist vonnöten, da eine Klassifizierung, die nicht perfekt ist, eine Verschlechterung des entstehenden Transkripts zur Folge haben kann.

Um die Klassifikationsmodelle zu trainieren wird der öffentliche Datensatz MUSAN genutzt, welcher Musik, Sprache und Störgeräusche unterschiedlicher Art enthält. Die letztendliche Leistung des Segmenters wird durch einen Vergleich der Transkriptqualität einer ASR mit vorgeschaltetem Segmenter sowie derselben ASR ohne Segmenter beurteilt. Dabei ersetzt der Segmenter alle Musik- und Störgeräuschsegmente mit Stille.

Die Ergebnisse zeigen, dass es möglich ist, mit einem Neuronalen Netz, welches eine Architektur von 30x20x10 hat, Frames der Größe 10 Millisekunden mit einer Genauigkeit von 87% zu klassifizieren. Dabei wird ein zeitlicher Kontext von 130 Millisekunden genutzt, was zu einer Latenz von 70 Millisekunden führt. Durch einen zweiteiligen Glättungsalgorithmus ist es möglich eine Transkriptqualität zu erreichen, die genauso gut ist, wie wenn

alle Musiksegmente von Hand entfernt werden. Dieser Algorithmus erzeugt eine Latenz von nur 200 Millisekunden. Im ersten Schritt wird dabei der Median von den Klassen der benachbarten Frames im Umkreis von 200 Millisekunden genutzt, um das aktuelle Frame neu zu klassifizieren. Im zweiten Schritt wird sichergestellt, dass ein Klassenwechsel von einem Frame zum nächsten im Falle von Musik bzw. Störgeräusch nur möglich ist, wenn die Mehrheit der letzten 3 Sekunden als Musik bzw. Störgeräusch klassifiziert wurde.

Contents

1 Introduction

Automatic speech recognition (ASR) systems generally respond poorly to music and noise input. The recognition delay increases due to a missing likely interpretation of the audio. The number of insertions increases in those segments due to falsely detected speech. Filtering those segments out is desirable to increase the accuracy of the ASR and to improve the average recognition speed. This is especially important for real-time systems, where a small delay is required, and processing power is generally limited. A delay of less than 5 seconds for the speech recognition is considered to be real-time, as humans perceive a caption delay of less than 5 seconds as acceptable [Las+12].

This work proposes and evaluates a segmentation algorithm to discriminate speech, music and noise from an audio stream, which can be used as a preprocessing step for an online ASR. In this context online refers to stream decoding. Music and noise segments are replaced by silence and therefore the ASR does not need to spend valuable computation time on those segments, and does not produce wrong transcripts for those music and noise segments. As the ASR already needs a couple of seconds create the transcript, the delay caused by the segmentation algorithm needs to be small (< 0.5 seconds) in order to satisfy the real-time constraint.

The segmentation uses a two step approach, consisting of classification and smoothing. Two machine learning models are investigated to classify audio frames into speech, music and noise: A multilayer perceptron and a support vector machine. Features based on Mel frequency cepstral coefficients and the zero-crossing rate are used as input for the classification. Different model parameters, as well as feature extraction parameters, such as frame size and frame context are evaluated regarding accuracy and induced latency.

The second step consists of a smoothing algorithm, which smoothes the classified frames to create segments of certain audio types and removes small misclassifications. Different smoothing parameters are compared. This step is necessary as using an imperfect classifier can actually lead to a decrease in transcript quality. The transcript quality is measured by the word error rate (WER).

To train and evaluate the machine learning models the MUSAN [SCP15] dataset is used, which is a publicly available audio dataset containing music, speech and noise. The end-to-end performance is evaluated by comparing the resulting transcript quality of the ASR with results of the ASR having the segmenter as preprocessing step, replacing music and noise with silence.

The results show that a classification model consisting of a multilayer perceptron with an 30x20x10 layer architecture is efficient at classifying 10 millisecond audio frames taking a temporal context of 130 milliseconds into account, with an accuracy of 87%, while maintaining a latency of 70 milliseconds. The results additionally show a two step smoothing algorithm to achieve good end-to-end performance results with the transcript quality en-par with the manual removal of all music segments while maintaining a small latency of 200 milliseconds. In the first step the smoothing algorithm replaces each frame with the mode of the adjacent frames withing a neighbourhood of 400 milliseconds. The second step assures that a class switch of the current frame to music / noise is only performed when the majority of audio frames in the last 3 seconds have been classified as music / noise.

2 Background

This chapter is supposed to give an insight to the background required for this thesis. First of all an overview of automatic speech recognition is presented, followed by a section about audio segmentation. The next section presents a short introduction into hidden Markov models, which are used in the field of automatic speech recognition, as well as audio segmentation. The two following sections present classification models. First an introduction in support vector machines is presented, then the concept of artificial neural networks is explained. In the last section two audio features which are used in this work are presented.

2.1 Automatic Speech Recognition

Automatic speech recognition is a subfield of spoken language processing focusing on the transcription of spoken language into word sequences. It has various applications, such as dictation, spoken language interfaces, entertainment systems, subtitle creation and aiding disabled people.

There are multiple difficulties in speech recognition, such as the vocabulary size, speaking style, speaker mode and signal quality. Applications like digit recognition for voice based calling achieve a higher accuracy then dictation, since the vocabulary size is much smaller. Recognizing isolated words is simpler than recognizing continuous speech. In continuous speech words are pronounced differently, depending on the words before and after. Whether the continuous speech is read or spontaneous makes a big difference in the recognition. Difficulties in spontaneous speech include the high variation of the speech rate, the use of dialect, discontinued words, repetitions of the same word, increased articulatory sounds like breathing and smacking sounds, bad or incorrect pronunciation and grammatically incorrect constructs. The quality of the signal, which suffers from background sounds like music or muttering, can have a big impact on the recognition accuracy [Ada10; Rog05].

A typical automatic speech recognition system consists of signal processing, decoding and adaptation. Feature vectors are extracted from the audio signal and then fed to the decoder, which generates word sequences that have the maximum posterior probability given the feature vectors. This probability is usually based on acoustic and language models. An Acoustic model refers to the representation of knowledge about phonetics, acoustics, environment variability, microphone type and differences among speakers like gender and dialect. It is usually based on hidden Markov models (see section 2.3 for an introduction into HMMs). A language model includes the possible vocabulary, possible semantics, the likelihood of a word in a certain context and how likely different word sequences are. The adaptation component modifies the models with the aim to represent the current environment situation, so that the performance can be improved [JM09, pp. 5].

To find likely word sequences in a reasonable time, the Viterbi beam search is commonly used. The beam search uses a breadth-first style search, but expands only nodes that are likely to succeed at each level, pruning the unlikely paths. For the beam search to be efficient, it is important that for a given observation there are word sequences which are more likely than others. Otherwise pruning cannot be applied safely since there are no nodes which are more promising than others, and all need to be traversed. In practice this leads to delay spikes for the transcription output of those word sequences, which also influence the transcription delay of the following words.

Noise and music as audio input lead to the absence of likely word sequences, and therefor the beam search causes high delays. By filtering out the noise and music segments in audio the corresponding delay spikes are removed, and the average delay is reduced.

This section is based on [HAH01], [JM09] and [Rog05].

2.2 Audio Segmentation

Segmentation of audio into multiple audio classes has a broad spectrum of usage: For instance, it is used in audio coding to switch between different music and speech coders based on the audio signal [Ram99]. Meta data about the content of an audio file can be automatically created [MB03]. Spoken language in radio broadcast can be detected to switch to another channel, in order to only listen to music [Sau96]. Furthermore other tasks, such as video segmentation can be aided by the segmentation of the corresponding audio [BW98]. As this work shows the performance of automatic speech recognition can be improved by filtering out non-speech segments. This could also be applied to improve automatically generated subtitles for videos.

The segmentation task can be split into two types: Offline and online segmentation. In the case of offline segmentation the segmentation is performed on complete audio files, and the process is not time critical. Pre- and post-processing steps can be applied to increase the segmentation accuracy. Temporal information can be used extensively to make segmentation decisions. When performing online segmentation there is only limited temporal information, segmentation decisions are made in real-time based on the incoming audio stream. As this work focuses on online segmentation the offline segmentation is not further investigated.

When segmenting online the audio stream is typically split into small frames from which feature vectors are extracted. Multiple feature vectors form a (sometimes overlapping) window, which is then classified by a classification model. The most commonly used models are based on Gaussian mixture models (GMMs) and hidden Markov models. The stream of classifications is usually smoothed within a small context to remove small misclassifications, forming the final segmentation.

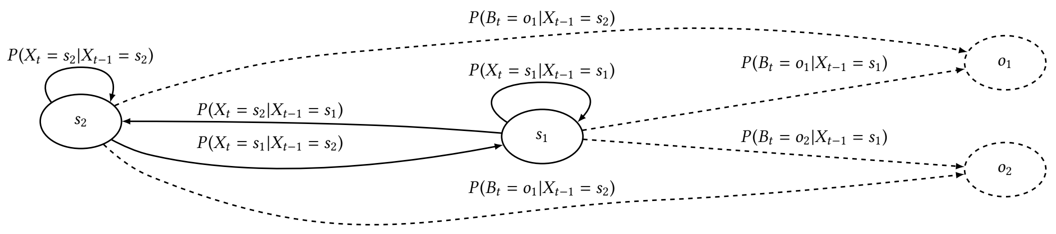
$P(B_t = o_1|X_{t-1} = s_2)$
$P(X_t = s_2|X_{t-1} = s_2)$
$P(X_t = s_1|X_{t-1} = s_1)$
$P(B_t = o_1|X_{t-1} = s_1)$
o_1
s_2
$P(X_t = s_2|X_{t-1} = s_1)$
s_1
$P(X_t = s_1|X_{t-1} = s_2)$
$P(B_t = o_2|X_{t-1} = s_1)$
o_2
$P(B_t = o_1|X_{t-1} = s_2)$

Figure 2.1: HMM with two hidden states s_1, s_2 and two possible outputs o_1, o_2. A solid edge indicates the probability of a state change from s_i to s_j, whereas a dashed edge indicates the probability of observing the output o_i given that the state is s_j.

2.3 Hidden Markov Model

This section gives an short overview of hidden Markov models (HMMs), based on [HAH01; Rab89]. Only HMMs with a discrete state space are discussed. A HMM is a statistical method to characterize hidden states based on observations.

Let $S = \{s_1, \cdots, s_m\}$ be a set of symbols, and $X = \{X_1, \cdots, X_n\}$ be a discrete sequence of random variables of S where the Markov assumption holds, which states that each random variable X_t depends only on the preceding value:

$$P(X_t|X_{t-1}, \cdots, X_1) = P(X_t|X_{t-1})$$

The sequence is then called *Markov chain*. By interpreting S as set of states, the probability $P(X_t = s_i|X_{t-1} = s_j)$ can be seen as the probability of being in state s_i at time t, given that the state was s_j at time $t - 1$. When including the probabilities $\pi_i = P(X_1 = s_i)$, stating that the initial state is s_i, the chain is called *observable Markov model*.

The *hidden Markov model* is an extension of the observable Markov model, where the states are note directly observable. Instead the output of states can be observed. Let $B = B_1, \cdots, B_n$ be a sequence of random variables in $O = \{o_1, \cdots, o_d\}$, where O is the set of output symbols. The probability of observing the output o_j given the hidden state s_i at time t is

$$P(B_t = o_j|X_t = s_i).$$

A simple example HMM can be seen in Figure 2.1.

The parameters of an HMM can be trained with the Baum-Welch algorithm, also known as forward-backward algorithm. In each of multiple iterations the new parameters are modified to maximise the likelihood of the observation given the old parameters. This algorithm converges against a local maximum in the parameter space [Rab89, pp. 264]. To get the most likely sequence of hidden states given a sequence of states (called decoding problem), the Viterbi algorithm can be used, which is based on dynamic programming.

2.4 Support Vector Machine

In its basic form a support vector machine is a binary classifier. Given some data points from two classes in the space H of dimension d, the SVM aims to separate those data points with a $d-1$ dimensional hyperplane h. From all possible separating hyperplanes the one that has the largest margin between the two classes is chosen. To classify non-linear separable classes, kernel functions can be applied, which transforms the feature space into another with possibly higher dimension, where the classes are more likely to be linearly separable.

The content of this section is based on [BH03] and [DHS00].

2.4.1 Maximal Margin Hyperplane

Let $S = ((x_1, y_i), \cdots (x_n, y_n))$ be the linearly separable training data, with $x_i \in H$ and $y_i \in \{-1, 1\}$ being 1 if x_i belongs to class A, and -1 if x_i belongs to class B. Then the hyperplane h with maximal margin between the samples of the two classes is searched. h can be defined through arbitrary $w \in H$, and $b \in \mathbb{R}$ as

$$h = \{x \in H | w \cdot x - b = 0\}$$

with the margin being the space between the two hyperplanes h_{-1} and h_1.

$$h_{-1} = \{x \in H | w \cdot x - b = -1\}$$
$$h_1 = \{x \in H | w \cdot x - b = 1\}$$

The geometrical distance between h_{-1} and h_1 is

$$d = \frac{2}{||w||}$$

as seen in Figure 2.2 for $H = \mathbb{R}^2$. So in order to maximize the margin, $||w||$ or equivalently $\langle w, w\rangle$ must be minimized, while each point x_i needs to be outside the margin and on the right side of the margin to not be misclassified. This is enforced by the following constraint:

$$\forall i \in \{1, \cdots, n\} : y_i(\langle w, x_i\rangle - b) \geq 1$$

This leads to the following convex optimization problem to find the maximal margin hyperplane:

$$\text{minimize}_{w,b} \langle w, w\rangle$$
$$\text{subject to } \forall i \in \{1, \cdots, n\} : y_i(\langle w, x_i\rangle - b) \geq 1$$

The classification decision rule is then given by $f(x) = sgn(\langle w, x\rangle + b)$.

Dual representation By forming the Lagrangian dual problem [BV04] the optimization problem can be efficiently solved by quadratic programming algorithms. The dual problem is:

$$\text{maximize}_\alpha \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i, x_j\rangle$$

$$\text{subject to } \forall i \in \{1, \cdots, n\} : \sum_{i=1}^{n} y_i \alpha_i = 0 \wedge \alpha_i \geq 0$$

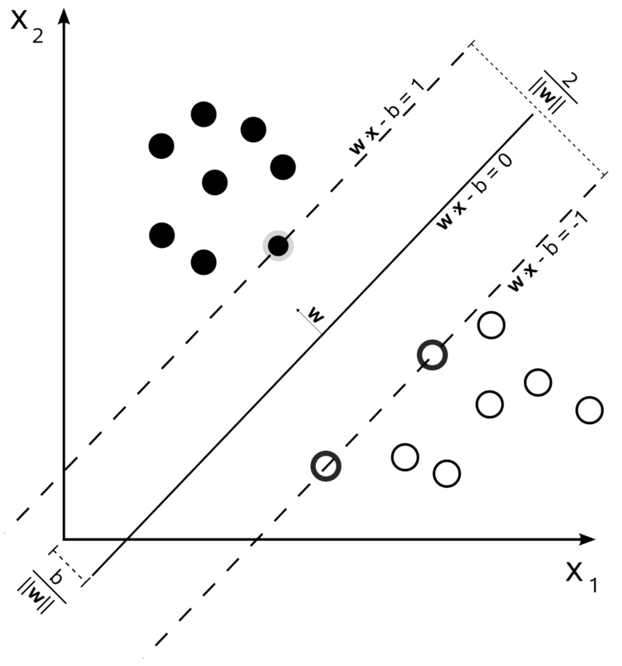

Figure 2.2: Example of a maximal separation hyperplane h defined by w, b in a 2 dimensional space (from [Wik11]).

Let α^* be the solution to this problem. Then the weight vector of the maximal margin hyperplane is $w^* = \sum_{i=1}^{n} y_i \alpha^* x_i$, and b^* can be calculated by solving the Karush-Kuhn-Tucker conditions, which state that the optimal solution must satisfy

$$\forall i \in \{1, \cdots, n\} : \alpha_i^* \left(y_i \left(\langle w_i^*, x_i \rangle + b^* \right) - 1 \right) = 0.$$

The corresponding decision rule is

$$f(x) = sgn(\sum_{i=1}^{n} y_i \alpha_i^* \langle x_i, x \rangle + b^*)$$

where $\alpha_i, i = 1, \cdots, n$ are the Lagrange multipliers. The Karush-Kuhn-Tucker conditions also imply that for the optimal solutions α^*, w^*, b^* only the α_i^* of the points x_i, which are on the margin border are non-zero. Only those are needed to calculate the decision. The corresponding points are therefore called support vectors. Let the set of the indices of those x_i be defined as SV. The decision rule can then be shortened to:

$$f(x) = sgn(\sum_{i \in SV} y_i \alpha_i^* \langle x_i, x \rangle + b^*)$$

2.4.2 Soft-margin

A few noisy or error prone training points can greatly affect the margin and thus can make the data non-linearly separable. To make the decision hyperplane more robust to noisy and erroneous data slack variables ξ_i are introduced, which allow training points to be on the wrong side of the decision boundary. This is achieved by modifying the constraint:

$$\forall \in \{1,\dots,n\} : y_i(\langle w, x_i\rangle - b) \geq 1 - \xi_i \wedge \xi_i \geq 0$$

To limit the amount and severity of margin violations the slack variables ξ_i are added to the optimization problem as error terms

$$\begin{aligned}&\text{minimize}_{w,b,\xi}\langle w, w\rangle + C\sum_{i=1}^{n}\xi_i^2\\ &\text{subject to } \forall i \in \{1,\dots,n\} : y_i(\langle w, x_i\rangle - b) \geq 1 - \xi_i \wedge \xi_i \geq 0\end{aligned}$$

with C being a parameter to control the tradeoff between the margin size and the classification errors.

Dual representation Similar to the last section, the dual optimization problem can be formed as:

$$\begin{aligned}&\text{maximize}_{\alpha}\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j(\langle x_i, x_j\rangle + \frac{1}{C}\delta_{ij})\\ &\text{subject to } \begin{array}{l}\sum_{i=1}^{n} y_i\alpha_i = 0\\ \alpha_i \geq 0\end{array} \quad \text{for } i = 1,\cdots,n\end{aligned}$$

with δ_{ij} being the Kronecker delta, which is 1 if $i = j$ and 0 otherwise.

2.4.3 Kernel Trick

To separate non-linear separable data, the data can first be transformed into a more powerful feature space F and then be linearly separated. For the transformation a mapping $\phi : H \rightarrow F$ is needed. The dual decision rule can be calculated using only the inner product of test and training points. Similarly solving the dual optimization problem can be evaluated using only the inner product between training points.

$$f(x) = sgn(\sum_{i\in SV} y_i\alpha_i^*\langle\phi(x_i), \phi(x)\rangle + b^*)$$

This makes it possible to implicitly define the mapping ϕ and the feature space F by defining the inner product $K : H \times H \rightarrow F$, $K(a, b) = \langle\phi(a), \phi(b)\rangle$ as a function of the original feature space. Such a function is called *kernel function*. Thus the decision rule can be written as:

$$f(x) = sgn(\sum_{i\in SV} y_i\alpha_i^* K(x_i, x) + b^*)$$

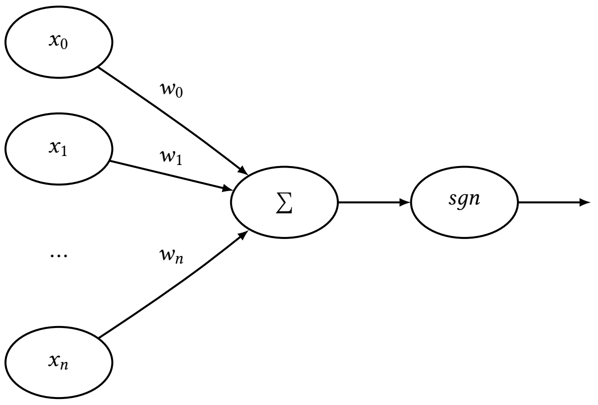

Figure 2.3: Graphical representation of a perceptron with weights $w_0, \cdots, w_n$, input values $x_0, \cdots, x_n$ and the sign function as activation function.

The domain of the kernel function and the function K itself can be arbitrary, as long as it is possible to prove that the implied mapping ϕ exists and is well defined. Kernels used commonly in SVM based classification include

$$K(a,b) = \langle a, b \rangle \qquad \text{(linear)}$$
$$K(a,b) = (\langle a, b \rangle + c)^d \qquad \text{(polynomial)}$$
$$K(a,b) = e^{-\frac{\|a-b\|^2}{2\sigma^2}} \qquad \text{(radial basis function)}$$
$$K(a,b) = tanh(\gamma\langle a, b \rangle + r) \qquad \text{(sigmoid)}$$

where all undefined variables are parameters for the corresponding kernel.

2.5 Artificial Neural Network

An *artificial neural network* (ANN) is a general, practical and robust approach at approximating a target function. ANNs are inspired by biological learning systems consisting of many interconnected neurons, which is a highly parallel process. This parallelism allows the use of highly parallel computer systems to train and evaluate ANNs. The backpropagation learning algorithm makes the model robust against errors and noise in the training data.

2.5.1 Perceptron

The most primitive type of ANNs is a *perceptron*. It takes a vector $\tilde{X} = (x_1, x_2, \cdots, x_n) \in \mathbb{R}^n$ as input and has some internal weights $W = (w_0, w_1, \cdots, w_n) \in \mathbb{R}^{n+1}$. Its output function $o(\tilde{X})$ is a linear function, that returns 1 for values above a certain threshold (defined by w_0), and -1 otherwise.

$$o(\tilde{X}) = \begin{cases} 1 & w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n > 0 \\ -1 & \text{otherwise} \end{cases}$$

Defining $x_0 = 1$ and $X = (x_0, x_1, \cdots, x_n)$, the output function can be simplified to:

$$o(X) = sgn(W \cdot X) = sgn(\sum_{i=0}^{n} w_i x_i)$$

A graphical representation of a perceptron can be seen in Figure 2.3. Each weight w_i represents the contribution of the input x_i to the output o. How well the perceptron models the target function t depends on these weights. The space containing all possible output functions is called *hypothesis space*. It is important that the hypothesis space contains a function that models the target function sufficiently close. The hypothesis space of a single perceptron contains only linear functions and is therefore quite limited. A classical example for the restriction of the perceptron is the XOR function $x_1 \oplus x_2$, which can not be separated by a linear function.

A perceptron is as powerful as a linear support vector machine. The difference between perceptrons and linear SVMs is mainly the way they are trained. As seen in section 2.4 SVMs are trained by solving a quadratic minimization problem under some constraints, while perceptrons are trained by gradient descent [CB04].

Activation Function To be able to use backpropagation to train multiple perceptrons as seen later, the output function o of a perceptron needs to be differentiable. Therefore o needs to be continuous. When redefining the output function as $o(X) = \sigma(\sum_{i=0}^{n} w_i x_i)$, different activation functions σ can be used. Possible activation functions are:

$$\sigma(y) = \begin{cases} -1 & y \leq 1 \\ y & -1 < y < 1 \\ 1 & y \geq 1 \end{cases} \qquad \text{(linear)}$$

$$\sigma(y) = \frac{1}{1 + e^{-y}} \qquad \text{(sigmoid)}$$

$$\sigma(y) = tanh(y) \qquad \text{(hyperbolic tangent)}$$

Multiple layers of neurons with a linear activation function can only produce a linear output function. To produce a good approximation of a target function often a highly nonlinear function is needed. With a nonlinear activation function a multilayer perceptron can produce a nonlinear output function. This can be compared to the kernel functions of support vector machines, which allow nonlinear decision functions. Usually the sigmoid function σ is used, since its derivative can be computed easily as $\frac{d\sigma(y)}{dy} = \sigma(y) \cdot (1 - \sigma(y))$ and the function is nonlinear.

Training by use of Gradient Descent The weights of a single perceptron can be trained by using gradient descent. Gradient descent is an optimization algorithm used in order to find a local minimum of a function by taking the gradient at the current position and shifting the weights in the opposite direction, until a minimum is found. In general the function is not convex, and only a local minimum is found.

To apply gradient descent a function (usually called *loss function*) representing the quality of the prediction for a certain set of weights $W = (w_0, \dots, w_n)$ is needed. One possible loss function is the squared error over all predictions for the training data D. Let t_d be the label of the input vector $d = (x_0, \dots, x_n) \in D$, and $o(d) = \sigma(\sum_{i=0}^{n} w_i x_i)$ be the predicted label. The squared error loss function is then defined as:

$$loss(W) = \frac{1}{2} \sum_{d \in D} (t_d - o(d))^2$$

The gradient of *loss* consists of the partial derivatives with respect to w_i:

$$\nabla loss(W) = (\frac{\partial loss}{\partial w_0}, \frac{\partial loss}{\partial w_1}, \cdots, \frac{\partial loss}{\partial w_n})^T$$

Each partial derivative can be calculated by:

$$\begin{aligned}
\frac{\partial loss}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o(d))^2 \\
&= \sum_{d \in D} (t_d - o(d)) \frac{\partial}{\partial w_i} (t_d - o(d)) \\
&= \sum_{d \in D} (o(d) - t_d) \cdot \frac{\partial o(d)}{\partial w_i}
\end{aligned}$$

The weights can now be updated using the following rule:

$$\begin{aligned}
\triangle w_i &= -\alpha \sum_{d \in D} (o(d) - t_d) \cdot o_i'(d) \\
\hat{w}_i &= w_i + \triangle w_i
\end{aligned}$$

where $o_i'(d)$ is $\frac{\partial o(d)}{\partial w_i}$ and α is called *learning rate*, since the parameter affects how big the gradient descend steps are. If α is too small, the algorithm converges very slowly towards a (possibly only local) minimum. If it's to large, minima can be overstepped. To overcome this problem, a dynamic α can be used.

Stochastic Gradient Descent For a large dataset of size n gradient descent takes a long time, since for each step and weight a sum with n summands has to be calculated. To speed this up, stochastic gradient descent (SGD) can be used. This method considers only one data point to calculate the stochastic gradient, the idea being to approximate the gradient descent.

$$\triangle w_i = -\alpha(o(d) - t_d) \cdot o'(d)$$

The smaller the parameter α is, the closer the SGD method approximates the true gradient descent. As a tradeoff *batch stochastic gradient descent* can be used, which updates the weights in small batches B of data.

$$\triangle w_i = -\alpha \sum_{d \in B} (o(d) - t_d) \cdot o'(d)$$

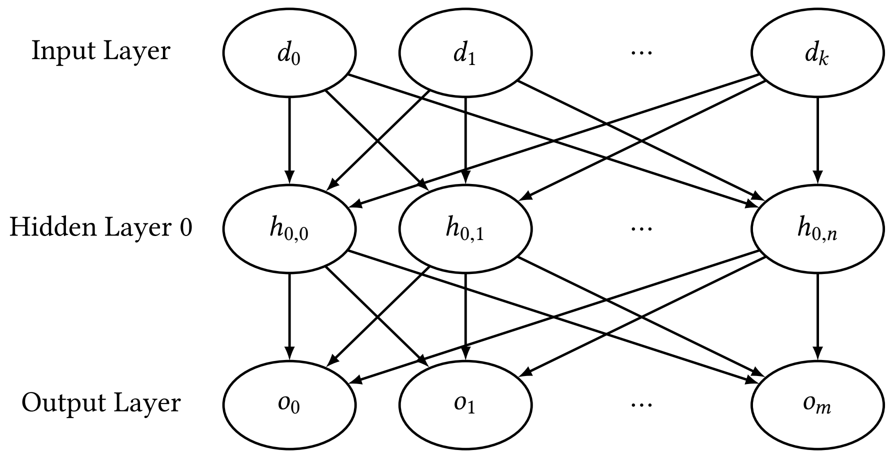

Figure 2.4: A MLP with one hidden layer. The input layer consists of k nodes, the hidden layer of n nodes, and the output layer of m nodes.

2.5.2 Multilayer Perceptron

A multilayer perceptron (MLP) contains multiple layers, including an input layer x, an output layer y and at least one hidden layer. A *hidden layer* h_i consists of a number of neurons, where the input of a neuron is the output of all nodes in layer h_{i-1}. In contrast to a single perceptron, a MLP with the sigmoid activation function can approximate any continuous function $f : \mathbb{R} \rightarrow \mathbb{R}$ with just one sufficiently large hidden layer [HSW89]. However the training algorithm is not guaranteed to find the right parameters to represent the function [IC16]. An increasing number of hidden layers can increase the generalisation with respect to the number of parameters as shown in [Goo+13].

Figure 2.4 shows the graphical representation of a MLP with one hidden layer. The evaluation and training can be realised with forward and backward propagation. The following notation is used to explain the forward and backward propagation:

$$\begin{aligned}
in_i &= \text{ input of node } i \\
o_i &= \text{output function of node } i \\
out_i &= o_i(in_i) = \text{ output of node } i \\
up(i) &= \text{the set of nodes that are in the layer above } i \text{ (closer to the input)} \\
down(i) &= \text{the set of nodes that are in the layer below } i \text{ (closer to the output)} \\
w_{ij} &= \text{the internal weight of node } j\text{, which is applied to the output of node } i
\end{aligned}$$

Forward Propagation To calculate the output of the MLP forward propagation is used. For simplicity this is shown with one hidden layer. Let L be the set of input nodes, d_l the input value of node $l \in L$, H the set of hidden nodes, and N the set of output nodes. The output of each node $n \in N$ can be calculated by forward propagation as:

$$o_n(X) = \sigma(\sum_{j \in H} w_{jn} \cdot \sigma(\sum_{i \in L} w_{ij} d_i))$$

Backward Propagation The backward propagation algorithm trains the weights of the MLP, using an approach similar to stochastic gradient descent. For every data point a loss function is calculated and the weights are updated in the opposite direction of the derivative of the loss function. This process is repeated until a certain condition is met.

The MLP can have multiple output units y, so the loss function needs to be modified:

$$loss_d(W) = \frac{1}{2} \sum_{k \in y} (t_{dk} - o_k(d))^2$$

$$\begin{aligned} \frac{\partial loss}{\partial w_{ij}} &= \frac{\partial loss}{\partial in_j} \frac{\partial in_j}{\partial w_{ij}} = \delta_j \frac{\partial}{\partial w_{ij}} \sum_{k \in down(j)} w_{kj} \cdot out_k \\ &= \delta_j \cdot out_i \end{aligned}$$

Where the calculation of $\delta_j = \frac{\partial loss}{\partial in_j}$ depends on the type of layer the node j belongs to. For j being a node in a hidden unit δ_j can be calculated as:

$$\begin{aligned} \delta_j = \frac{\partial loss}{\partial in_j} &= \sum_{k \in up(j)} \frac{\partial loss}{\partial in_k} \frac{\partial in_k}{\partial in_j} = \sum_{k \in up(j)} \delta_k \cdot \frac{\partial}{\partial in_j} \sum_{l \in down(k)} w_{lk} \cdot o_l(in_l) \\ &= \sum_{k \in up(j)} \delta_k \cdot w_{jk} \cdot o'_j(in_j) \end{aligned}$$

And for j being a output unit δ_j can be calculated as:

$$\begin{aligned} \delta_j = \frac{\partial loss}{\partial in_j} &= \frac{\partial}{\partial in_j} \frac{1}{2} \sum_{k \in \mathrm{N}} (t_k - o_k(in_k))^2 = \frac{\partial}{\partial in_j} \frac{1}{2} (t_j - o_j(in_j))^2 \\ &= (o_j(in_j) - t_j) \cdot o'_j(in_j) \end{aligned}$$

Similar to the gradient descent the weights can now be updated by the following rule, where α is again the learning rate:

$$\begin{aligned} \triangle w_{ij} &= -\alpha \cdot \delta_j \cdot out_i \\ \hat{w}_{ij} &= w_{ij} + \triangle w_{ij} \end{aligned}$$

Since the search space contains multiple local minima in general, the algorithm most likely won't find the global minimum, but will converge to a local minimum. Despite this, backpropagation is in practice a good algorithm to approximate functions. [Mit97]

2.6 Sound Features

Choosing features that represent the sound signal while being of low dimensionality is important in the task of audio classification. Multiple features are used in audio classification. The two most common features are Mel frequency cepstral coefficients and the zero-crossing rate, which are therefore presented.

2.6.1 Mel Frequency Cepstral Coefficients

MFCCs are short-term spectral-based features. They are commonly used in speech recognition, since they can represent the speech amplitude spectrum compactly. There are multiple similar definitions of MFCCs, so this section focuses on the definition of [HAH01, pp. 313] and [Log+00].

Splitting into frames The first step to calculate the MFCCs is dividing the signal into short frames. This is usually done by applying the Hamming window function to fixed intervals of 20 ms. An aliased cepstral feature vector $(x_0, \cdots, x_{N-1})^T$ is created for each frame.

Discrete Fourier Transformation The discrete Fourier transformation is taken from each feature vector, yielding N complex spectral components $X_0, \cdots, X_{N-1}$ with

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{\frac{-2\pi ikn}{N}}, k = 0, \ldots, N-1$$

Mel-scaling and energy bin creation The spectral components are reduced to M spectral bins. The bins are distributed according to the Mel scale, since lower frequencies seem to be more important than higher frequencies. The Mel scale is a perceptual scale, mapping an actual frequency to pitch as perceived by human listeners. The conversion of x Hz into $B(x)$ mel can be calculated by:

$$B(x) = 1125 \cdot ln(1 + \frac{x}{700})$$
$$B^{-1}(x) = 700 \cdot (e^{\frac{x}{1125}} - 1)$$

Let l_f and l_h be the lowest and highest possible frequency in Hz, and F the sampling frequency in Hz. The boundary points $f_0, \cdots, f_M$ of the bins are calculated as:

$$f_m = \frac{N}{F} \cdot B^{-1}\left(B(l_f) + m\frac{B(h_f) - B(l_f)}{M+1}\right)$$

A triangle filter bank with M filters $H_1, \cdots, H_M$ is used to calculate the average power spectrum around each boundary point f_m. The filters are defined as:

$$H_m[k] = \begin{cases} \frac{k-f_{m-1}}{f_m-f_{m-1}} & f_{m-1} \leq k \leq f_m \\ \frac{f_{m+1}-k}{f_{m+1}-f_m} & f_m < k \leq f_{m+1} \\ 0 & \text{otherwise} \end{cases} \quad \text{, for } 0 \leq m < M$$

Now an energy bin can be created from the output of each filter. The logarithm is taken and the log-energy bins can be computed as:

$$S[m] = ln\left(\sum_{k=0}^{N-1} |X_k|^2 H_m[k]\right) \quad \text{, for } 0 \leq m < M$$

Discrete Cosine Transformation To decorrelate the overlapping bins a discrete cosine transformation is applied to the log-energy bins, forming M Mel frequency cepstral coefficients.

$$mfcc[j] = \sum_{m=0}^{M-1} S[m] \cdot cos\left(\pi j \frac{m+\frac{1}{2}}{M}\right)$$

2.6.2 Zero-crossing Rate

The zero-crossing rate measures the number of sign changes in the signal per window. It is defined as

$$zcr = \frac{1}{T-1} \sum_{t=1}^{T-1} f\{s_t s_{t-1} < 0\},$$

where s is the sound signal with the length of the window being T, and f being defined as

$$f\{X\} = \begin{cases} 1 & X \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Analysis of the zero-crossing rate offers a simple and low dimensional feature to describe the spectrum. However because of its simplicity it is not sufficient by itself as an accurate description of the spectrum. [Ked86].

3 Prior Work

In this chapter an overview over previous work focusing on segmenting audio is given. The literature focuses mostly on speech and music discrimination, with greatly varying delay. Different classification models as well as features are used to achieve the task. Models based on Gaussian mixture models are most commonly used. Some segmentation approaches apply a form of smoothing to increase the accuracy. Mostly the classification accuracy of audio frames is investigated, the actual performance increase of ASR systems is rarely measured. We are only aware of one previous work [Hec+13] that measures the word error rate. In contrast this work focuses on the improvement of online ASR systems. Additionally previous work focuses on delays of more than a second to obtain high accuracies, which is barely real-time applicable, while this work focuses on obtaining a small delay of less than 0.3 seconds, in order to provide a good real-time capability.

Unfortunately there is no general audio corpus on which the different models are compared. Recently the MUSAN dataset, containing speech, music and noise, has been published, which could serve as a general corpus for audio classification in future work [SCP15]. The datasets of the previous work vary greatly, making it difficult to compare results.

3.1 Classification Models

This section presents prior work that focuses on the frame wise discrimination of audio classes.

Harb and Chen [HC03] propose an algorithm that discriminates speech and music based on limited training data (80 seconds), in order to be able to retrain the classifier fast, so different audio conditions can be accommodated. Features are based on the Mel frequency spectral coefficients and extracted in 30 ms frames with an 20 ms overlap. The first order statistics are taken from the mean and variance of the frames withing a 0.2 second window and used as the feature vector. As classifier a Multi Layer Perceptron is used, which outperformed the also tested k-Nearest Neighbour algorithm by 7%. Tests showed the context-independent classification accuracy to be 93%, while the context-dependent classification accuracy was 96%.

The work [PT14] uses a deep architecture consisting of Restricted Boltzmann Machines in order to differentiate music and speech. The spectrogram of the signal and 13 MFCCs are compared as features. For the MFCC features the audio is split into 50 ms frames and the delta and double delta coefficients are added to the feature vector x_i, as well as the frames x_{i-1} and x_{i+1}. A confidence threshold where frames are left unclassified if all class probabilities are below a certain threshold is applied. A classification error between 8% (FFT) and 12% (MFCC) has been achieved without a confidence threshold. This could be reduced to between 4% (FFT) and 5% (MFCC) with a confidence threshold of 0.9 having between 10% (FFT) and 20%(MFCC) of unclassified frames.

In [MB03] four different features are compared for the task of classifying audio files. The possible main classes are classical music, popular music, speech, noise and crowd noise. The popular music class is further divided into 7 subclasses. The features are based on low-level signal properties, MFCCs, psychoacoustic features like roughness and sharpness, and an auditory model representation of temporal envelope fluctuations. Features are retrieved by taking 743 ms windows, each containing half-overlapping frames of size 23 ms. Quadratic discriminate analysis and Gaussian mixture models are used to classify the feature windows. The classifier with low-level signal based features reached an accuracy of 86%, with the other features it reached an accuracy of about 92%.

Saunders proposes an algorithm to discriminate speech from music in broadcast FM radio in [Sau96]. A multivariate-Gaussian classifier is used to classify 2.4 second windows consisting of features taken from 16 ms frames. The features are primarily based on energy based features and the zero-crossing rate (ZCR). Both feature types use the number of values below or above a statistic based threshold as features. Zero-crossing based features are created by counting the difference between the number of frames with a high ZCR and a low ZCR. An accuracy of 98% could be achieved by this algorithm.

3.2 Segmentation Models

In this section prior work is presented that investigates the segmentation of audio, commonly by relabeling audio frames based on the surrounding frames.

In the publication [PT05] an algorithm is proposed to split audio into silence, speech and music segments. First the audio is segmented, then the segments are classified. The segmentation is based on the Root Mean Square (RMS) of 20 ms signal frames, with a minimum segment length of 1 second. It causes a 3 second delay and is therefore near real-time applicable. A 97% detection probability of a segment change could be achieved, where the accuracy of the change instant was mostly within an interval of 0.2 seconds. The segments are then classified with multiple threshold tests based on RMS and zero-crossing. A classification rate of 95% has been reported for this method.

In [WGY03] a fast near real-time speech/music segmentation algorithm is presented. A novel feature called modified low energy ratio is used with a frame size of 20 ms and a window length of 1 second. The Bayes maximum a posteriori probability decision rule is used to classify the feature windows, achieving a classification accuracy of about 91%. A context-based post-decision approach to smooth the classification is presented, which starts a new segment only if 4 consecutive windows are not classified as the class of the current segment. Using this approach, the accuracy could be increased to almost 98%, with a dynamic delay between 1 and 4 seconds.

In the work [LZL03] audio is discriminated into five different audio classes: Silence, music, background sound, pure speech and non-pure speech. Multiple features based on MFCCs and perceptual features are used. The frame size is 25 ms with a window size of 1 second. Support Vector Machines (SVMs) are used to classify the feature windows. The different classes are arranged as leaf nodes in a binary tree with a SVM at each inner node to classify into two groups of subclasses. Smoothing is performed by three simple rules considering the window before and after the current window. Excluding the silence, the smoothed classifications reached an accuracy of 92%. Other models, namely K-Nearest Neighbour and Gaussian Mixture Model, are compared to the SVM classification, which performed worse. Smaller window sizes were tested, and a window size of 0.3 seconds yielded an accuracy of about 85%. The algorithm is real-time applicable, as the inherent delay is only two times the window size.

In [Hec+13] an algorithm to segment recorded telephone conversations into speech and non-speech is proposed. SVM and GMM based classifiers are compared. The features are based on MFCCs. The features for the SVM based method are taken from 16 ms frames with a frame shift of 10 ms. 15 adjacent frames are stacked and linear discriminant analysis is applied to form a feature vector. To smooth the classification, a step of erosion followed by a step of dilation is applied. Additionally speech segments with a distance of less than 0.5 seconds are merged. Recordings of multiple not so well investigated languages are used for training and testing. To evaluate the algorithm the WER of an ASR with this segmentation was compared to the performance of an ASR with manual segmentation. The relative WER was between a 2% increase compared to the manual segmentation (Tagalog) and a decrease by 1.2% (Pashto). The smoothing process of this work is inspired by [Hec+13].

In [Mal+00] a real-time speech music discrimination algorithm is proposed, with an inherent delay of 20 ms. A quadratic Gaussian classifier is used to classify audio frames. The features are based on Line Spectral Frequencies and higher order crossings. Features are retrieved from frames of size 20 ms. To smooth the classifications the immediate two preceding frames influence the current frame classification. The smoothing increased the performance by 5% to 10%. The reported accuracy is 78%. Using 50 frames as a 1 second window to be classified, an accuracy of 96% could be reached.

4 Method

This chapter describes the conception and realisation of the proposed method. In Section 4.1 the segmentation process is outlined. The following sections 4.2 to 4.4 describe the processing steps in detail, and state parameters that are getting evaluated in the next chapter.

4.1 System overview

The segmenter acts as a preprocessing step for the automatic speech recognition system. It consists of three steps as seen in Figure 4.1. The initial step is the feature extraction, in which the audio stream is split into small segments from which feature vectors are extracted. In the second step, the classification, the audio class to which a feature vector belongs to is predicted. To avoid small misclassified segments, the predictions are smoothed in the smoothing step. The audio stream is then segmented according to the smoothed predictions, and non speech segments are replaced with silence. The resulting audio stream is then used as input for the ASR which outputs the transcript.

4.2 Feature Extraction

Expressive spectral and temporal features are required to achieve high accuracy in audio discrimination. Mel frequency cepstral coefficients are used extensively in speech recognition in general [Log+00] , and have proven to be successful at discriminating speech and music as seen in previous work [MB03]. This work focuses on MFCC and Zero Crossing based features.

In online discrimination the length of the time window used for feature extraction needs to be small, in order to keep the inherent delay small. Humans need windows of about 200ms to easily achieve classification [HC03]. As the evaluation shows, a window length of 130 ms is sufficient to discriminate with an accuracy of 87.5%.

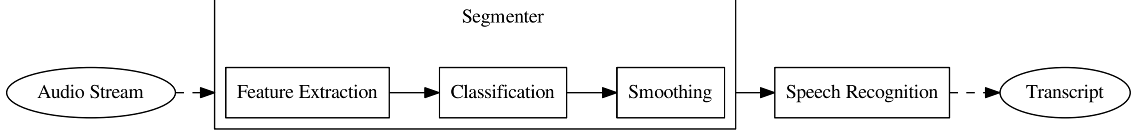

Figure 4.1: This figure shows the processing steps from audio to transcript. The output of the classification is evaluated, as well as the output of the speech recognition.

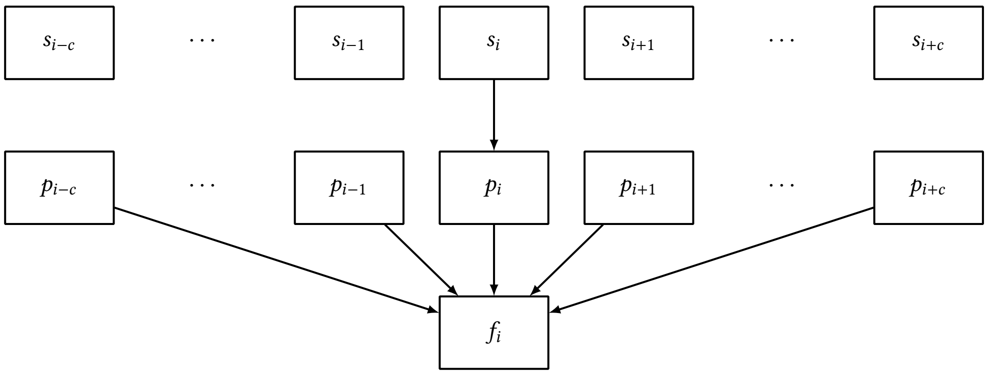

Figure 4.2: The extraction of feature vectors is shown in this figure. The i-th audio segment s_i of length $len(s)$ (ms) is used to compute the preliminary feature vector p_i. Statistic functions are applied to the adjacent vectors $p_{i-c}, \cdots, p_{i+c}$, where c is the feature context, to create the feature vector f_i.

The feature extraction process can be seen in Figure 4.2. Features are extracted by splitting the audio stream in frames s_i with a length of $len(s) = 10$ ms, and computing MFCC and Zero Crossing preliminary features vectors p_i from these frames. The number of MFCCs is denoted by m.

$$p_i = (mfcc_0, mfcc_1, \cdots, mfcc_{m-1}, zc)^T$$

Adjacent preliminary feature vectors are combined into the feature vector b_i, where $c = 6$ is the feature context.

$$b_i = (p_{i-c}, \cdots, p_{i+c})$$

By adding adjacent vectors, the window size is increased, and temporal change is taken into account. The resulting feature vectors $b_i, b_{i+1}, \cdots$ have a dimension of $b^d = len(p) \cdot (2c+1)$. To reduce the dimension, and thereby increasing classification speed, statistic functions are taken from the vector b_i to create the vector f_i.

$$\begin{aligned} f_i =& (mean(b_{i,0}), mean(b_{i,1}, \cdots, mean(b_{i,len(p)}), \\ & (variance(b_{i,0}), variance(b_{i,1}, \cdots, variance(b_{i,len(p)}))^T \end{aligned}$$

Having n different statistic functions, the dimension is reduced to $f^d = len(p) \cdot n$. To prevent early saturation of the neural net, each feature vector f_i is scaled to $[-1, 1]$. The delay caused by the feature extraction is $(1 + c) \cdot len(s)$. Computationally caused delay is neglected for the evaluation, since it depends on implementation and hardware.

Feature Parameter The following parameters are optimized in section 5.4.3, with the goal to achieve a good precision while maintaining a small delay: The length of audio frames $len(s)$, the number of MFCC buckets m, the feature context c, the combination of statistic functions in $\{mean, variance, stddev\}$ and the combination of features in $\{mfcc, zerox\}$.

4.3 Classification

The feature vectors are classified in either speech, music or noise. An accurate and fast classifier is required to accommodate for the real-time delay constraint. For this reason a multilayer perceptron is proposed as classification model. MLPs have been proven to be successful at discriminating speech and music as seen in [HC03]. As section 5.4.1 shows that an architecture with small hidden layers is sufficient, the computation overhead is quite small with about 1 ms on a single core of an AMD Opteron 6136 Processor with a clock speed of 2.4 GHz. A Support Vector Machine is used as comparison model. SVMs have been shown in previous work to be successful at discriminating speech and music [LZL03].

4.3.1 Multilayer Perceptron

The architecture of the Multilayer Perceptron can be seen in Figure 4.3. It consists of the input layer, three hidden $\{h_0, h_1, h_2\}$ layers, and the output layer. The input layer contains feature dimension f^d nodes, followed by the hidden layers consisting of $size(h_0) = 30$, $size(h_1) = 20$ and $size(h_2) = 10$ neurons. The output layer o consists of 3 neurons, one for each class. Each layer uses the sigmoid function as activation function. To retrieve class probabilities, the output of layer o is fed to the $softmax$ function.

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j \in \{0,1,2\}} e^{x_j}}$$

Where x_i is the weighted and biased output of the i-th neuron in layer o. As loss function the partial squared error is used.

Classification Parameter To increase the classification of the multilayer perceptron the following parameters are optimized in section 5.4.1. The learning rate, the number of nodes in the hidden layers, the batch size and the number of training steps. Since a small number of hidden nodes is shown to be sufficient, the number of hidden layers is not investigated. Additionally the small MLP architecture prevents overfitting from being a problem.

4.3.2 Support Vector Machine

A Support Vector Machine is used as comparison model. However the focus of this work is on the neural net based classification and therefore the SVM based model is only used to evaluate the classification, and not for the segmentation evaluation.

Since the decision surface of a support vector machine classifies into two class spaces, an additional algorithm for multi-class classification is needed. In this work the one vs all approach, as seen in [HL02] is used. In this approach one SVM classifier is used per class to fit the class against all other classes. For a feature vector the class where the corresponding classifier has the largest value of the decision function is chosen.

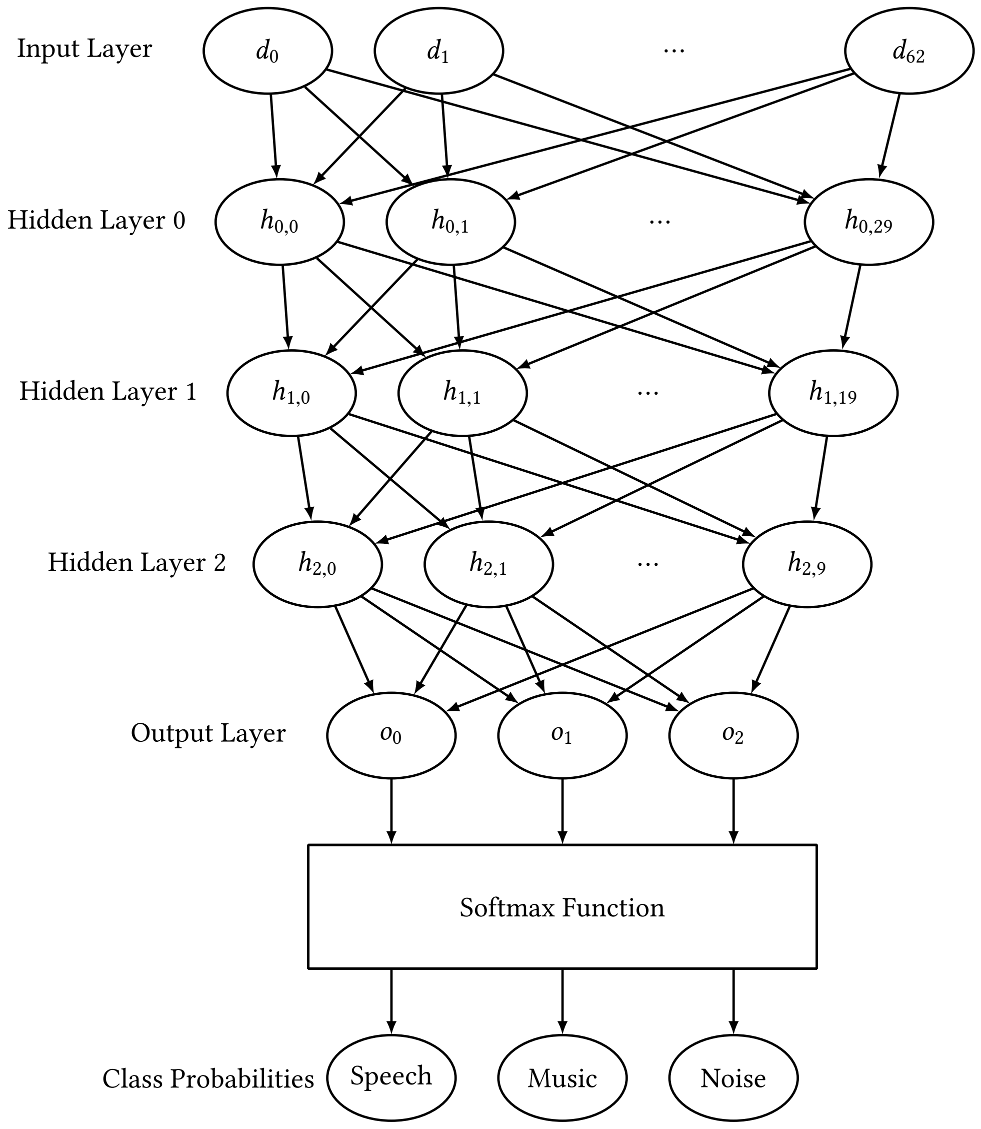

Figure 4.3: The architecture of the multilayer perceptron in use to classify the features. The input consists of a 63-dimensional feature vector. Three hidden layers h_0, h_1, h_2 are used, with layer h_0 containing 30, h_1 20 and h_2 10 nodes. The output of the output layer is fed to the softmax function, in order to retrieve class probabilities for each audio class.

Classification Parameter To find good model parameters, a random search is conducted. The following parameters are getting optimized in section 5.4.2: The penalty factor C of the error term, the gamma parameter of the kernel function, and the kernel being selected from $\{poly, rbf, sigmoid\}$.

4.4 Smoothing

Removing small segments within a speech section can greatly decrease the speech recognition performance. For this reason small segments classified as non-speech in a bigger speech segment should generally be relabeled to speech, as it is most likely that they have been misclassified. This is based on the assumption that speech and music segments are generally long. The smoothing process aims to remove those small misclassifications, and generate long continuous segments.

The process is split into four different steps. First a mode based filter, with a small context, is used to remove small misclassifications. The second and third step consist of erosion and dilation, as commonly used in image processing. This has been proposed to use in audio segmentation by [Hec+13]. Through erosion small speech segments are removed, dilation increases the size of the speech segments. The final step consists of changing current class only if the previous frame classifications support this change.

Mode Smoothing In this step the label l_i of the classified audio frame a_i is recalculated by taking the mode of the labels $\{l_{i-c}, l_{i-c+1}, \cdots, l_{i+c}\}$ where c is the mode context.

Erosion The erosion step removes small segments of speech or noise occurring in music, by relabeling the label l_i to music if a label in $\{l_{i-c}, l_{i-c+1}, \cdots, l_{i+c}\}$ is music. C being the erosion context. Noise is not used for erosion, since small noise segments are likely to occur, and there is no benefit in widening them.

Dilation After the erosion step is performed, the speech segments are dilated. This is done by relabeling l_i to speech if at least one label in $\{l_{i-c}, l_{i-c+1}, \cdots, l_{i+c}\}$ is speech. In this case c is the dilation context.

Minimum Change Support To prevent fast reoccurring class changes, the mean of the last min_x frames $\{a_{i-min_x}, a_{i-min_x+1}, \cdots, a_i\}$ needs to be class x, to change the label l_i to x. Each class x has its own min_x parameter, since the current classification needs to change faster for some classes than others. It is more important to get all speech frames, than to falsely classify music or noise as speech. Therefore min_{speech} should be small.

Smoothing Parameter To optimize the effectiveness of smoothing, multiple parameters can be tuned. The different contexts of mode smoothing, erosion and dilation can be tuned. By setting a context to zero the whole step is effectively disabled. This can be used to compare which steps are necessary and give the biggest performance boost. Generally small contexts are preferred, since they each contribute to the delay by $c \cdot len(frame)$

ms. min_{speech}, min_{music} and min_{noise} are optimizable parameters, where zero can be used as well to disable the step, however they do not contribute to the delay and can therefore be bigger. Section 5.4.4 focuses on the optimization of the smoothing parameters.

5 Experiments

This chapter describes the proposed experiments, and presents the evaluation of those experiments. Tools used to create the experimental environment are presented in section 5.2. The datasets on which the tests are performed are introduced in section 5.3. In section 5.4 the experiments and their results are presented.

5.1 Setup

This chapter describes the setup of the different kinds of experiments. For the ease of testing and implementation, the segmentation is done as an offline preprocessing step, while carefully adhering to the real-time constraints laid out in chapter 4. WAVE audio files are used as audio input. However, this does not affect the online capability of the segmentation, as seen in chapter 4.

Classification Setup The classification model is trained and tested on the MUSAN (section 5.3.1) corpus, the PBC dataset (section 5.3.2) is used as an out of domain test. All audio data has a sample rate of 16 kHz, uses one channel, and has a precision of 16 bit.

The MUSAN corpus is split into training, validation and test sets for training and evaluation. The split is based on audio files, and assigns 80% of each class in the corpus to the training set, and 10% each for the validation and testing set. The distribution of speech, music and noise is chosen to be even. Another possibility is to take a distribution more realistic to the real occurrence to where it is used. For example lectures would generally have a higher amount of speech. However the method aims to be applicable in multiple environments without retraining, so an even distribution approach is used. Additionally the segmentation part already favours speech and - to a certain extent - makes up for the even classification. This way the classification results are more comparable to other work by themselves. To get an even distribution, not all data can be used for training and evaluation. The amount of chosen audio for the training dataset is 250 minutes, for the validation and test dataset 25 minutes each. As evaluation criteria the classification accuracy and the inherent delay caused by the feature extraction is used. The accuracy is defined as:

$$accuracy = \frac{\text{number of correctly classified frames}}{\text{number of frames}}$$

The feature extraction induced inherent delay $d_{feature}$ is based on the frame size f_s and context f_c, additional computational delay is not considered since it is negligible. The inherent delay can be calculated as:

$$d_{feature} = f_s \cdot (f_c + 1)$$

Smoothing Setup Evaluating the quality of the smoothing parameters is not straight forward. One approach is to compare the resulting segmentation of an audio file to a manually created segmentation. However segment borders are perceived differently by different humans [Eri09]. The approach used in this work is to use the transcriptions created by the ASR as a quality measure. This has the advantage that the parameters chosen through this quality measure create segmentations that are favorable for ASR systems, even if they might be biased towards increasing speech segments.

In more detail: First the frames of the audio file are classified with the standard model from the classification setup. Then the classification is smoothed according to the smoothing parameters, and the resulting segmentation is used to modify the audio file by replacing noise and music segments with silence. The resulting audio file is fed to an automatic speech recognition system, and the transcript hypothesis is compared to the reference transcript by calculating the word error rate (WER). The WER is a commonly used metric to compare the performance of ASR systems. It is defined as:

$$WER = \frac{sub_{r \to h} + del_{r \to h} + ins_{r \to h}}{w_r}$$

where $sub_{r \to h}$ is the number of substitutions, $del_{r \to h}$ the number of deletions, and $ins_{r \to h}$ the number of insertions, to get from the reference transcript r to the hypothesis h, and w_r is the number of words in the reference r. Different smoothing parameter are compared by comparing the resulting word error rates of the transcribed segmented audio files. Additionally a baseline is used to show how well the smoothing performs compared to no segmentation at all.

Two audio files have been chosen on which the smoothing is evaluated: The audio file from the STC.TA test set, as well as the audio file from the STC.G test set. Both test sets are discussed in more detail in section 5.3.3. The STC.TA file consists of a TED talk including artificially added music, and is good to see how well the smoothing performance is when the music and speech well separated. The STC.G file consists of a guitar lecture and is a lot more challenging for the segmentation, since the transition between music and speech is not so clear, and sometimes overlapping.

Additionally to the WER, the delay caused by the smoothing is taken into account as a quality factor, since it is important for an online ASR system to have a small delay.

5.2 Tools

This section describes the main tools that have been used to create the experimental environment.

5.2.1 Python

Python is a high level programming language which is very popular for scientific computing. A wide range of scientific libraries are available in python, making it an ideal choice as the main language to implement the experiments of this work (in version 2.7.6).

5.2.2 scikit-learn

scikit-learn [Ped+11] is an open source machine learning library for the python programming language. It contains a wide range of state-of-the-art supervised and unsupervised machine learning algorithms, aiming to create an efficient yet easy to use library. The support vector machine implemented in this work is based on scikit-learn in version 0.17.0.

5.2.3 TensorFlow

TensorFlow [Mar+15] is an open source software library which was originally developed by Google for machine learning and deep neural network research. It has since grown to a more general numerical computation library based on data flow graphs. It features portability for computation on CPUs and GPUs for a wide variety of platforms. Additionally it provides an C++ interface as well as an Python interface. The neural networks in this work are implemented with TensorFlow (version 0.6.0).

5.2.4 openSMILE

openSMILE [Eyb+13] is an comprehensive audio feature extraction tool that can be used in real-time. It can use files in the Waveform Audio File Format as in- and output in addition to live sound recording and playback. Multiple general audio signal processing functions like windowing functions, fast Fourier transformation (FFT) and cepstrum are available. A vast number of features is provided, with the additional possibility to create feature summaries (statistical functions). Components can be easily connected via a configuration file. In this work openSMILE (in version 2.1.0) is used to retrieve audio features and create frame based statistics.

5.2.5 Janus Recognition Toolkit

The Janus Recognition Toolkit (JRTk) is an automatic speech recognition toolkit developed at the Interactive Systems Lab at Karlsruhe Institute of Technology and Carnegie Mellon University. The JRTK is implemented in C, and provides an object oriented Tcl/Tk script based environment. It contains various techniques for acoustic pre-processing, acoustic modeling and decoding. The ASR system used for the evaluation is based on the JRTk [Sol+01; Lav+97].

5.3 Datasets

The main dataset used for classification training and testing is MUSAN, as it is publicly available, and therefore is suitable for comparison among different work. Furthermore it provides a good range of audio sources, increasing the possibility of generalisation. As the PBC corpus contains only data from one specific source it is well suited as a out of domain test dataset. The available transcript makes the PBC corpus usable to test the benefit of segmentation. To evaluate the ASR performance the segmentation test corpus is used, which consists of multiple recordings, including transcripts.

5.3.1 MUSAN

MUSAN [SCP15] is a corpus containing speech, music and noise from various audio sources. It has been published on October 2015, and created with the aim to provide a standard corpus without copyright issues. The audio files are in the US Public Domain or under a Creative Commons license.

Speech The corpus contains about 60 hours of speech. From those 20 hours and 21 minutes are read speech from LibriVox, where each audio file is an entire chapter of a book. About half of the recordings are in English, the other half being from eleven other languages. The remaining 40 hours contain US government hearings, committees and debates, obtained from the Internet Archive and the Missouri Channel senate archives.

Music 42 hours and 31 minutes of the corpus consists of music from various sources. The music contains various genres from the following sources: Jamendo, Free Music Archive, Incompetech and HD Classical Music.

Noise Around 6 hours of the corpus is noise. The source of the noise is the Sound Bible and Free Sound. The noise contains technical noises, such as dial tones and fax machine noises, as well as ambient sounds like car idling, thunder, wind, footsteps, paper rustling, rain, animal noises, crowd noises.

5.3.2 PBC

The PBC Corpus originates from recordings at the Peninsula Bible Church in Silicon Valley. The Interactive Systems Lab at KIT has a pilot project at the church, with the goal of aiding deaf and deaf-mute people at understanding the church service. Multiple communication channels are used as aid, namely lip reading, weak acoustic signals for the near deaf, presentation charts (PowerPoint) and speech recognition with a small delay.

The corpus consists of multiple church service recordings, of which 158 minutes are speech, and 370 minutes are music. Noise is not explicitly classified. For the speech parts transcripts were crowd sourced, although the quality of the transcriptions is therefor not flawless it can still be used to compare different segmentation settings and setups.

5.3.3 STC

The segmentation test corpus consists of five test sets, each containing an audio file and a reference transcript.

STC.G - Guitar lecture This test consists of 8 minutes of a guitar lecture. The audio consists of frequent changes between guitar performance and explanation. Occasionally the guitar is played in the background while explaining.

STC.T - TED talk The second set consists of the TED talk "A garden in my apartment" from Britta Riley. It consists purely of 7.5 minutes of speech.

STC.TA - TED talk including artificially added music This set is based on the same TED talk as STC.T, but contains 2.5 minutes of artificially inserted music. The music is usually surrounded by short moments of silence.

STC.P - PBC recording A recorded church service from the PBC dataset is used for this test set. The whole recording is 89 minutes long, with about 48 minutes of speech, and 41 minutes of music. The music consists mostly of a choir accompanied by various instruments.

STC.PC - PBC recording, music cut out This set is based on the same recorded church service as STC.P, excluding all music, which has been cut out.

5.4 Evaluation

This section presents the evaluation results. First the MLP model is investigated, then the features are investigated. Afterwards the smoothing is evaluated, followed by an evaluation of the complete segmentation. As most evaluation sections focus on one specific part of the classification / smoothing, Table 5.1 presents a list of default feature extraction parameters that are used for the evaluation, if not stated otherwise.

Parameter	Value
statistics	mean, variance, stddev
features	MFCC, ZCR
MFCCs	20
frame size	10 ms
frame context	6

Table 5.1: Default parameters for the feature extraction.

5.4.1 Multilayer Perceptron Model

To find a MLP architecture that performs well on the given classification task, the following parameter have been tested with multiple values: The batch size, the learning rate, the number of nodes per hidden layer, and the number of training steps. The number of hidden layers has been chosen to be 3 , which proved to be sufficient. The activation function is also not investigated in more depth and has been selected to be the sigmoid function. A greedy search found the values seen in Table 5.2 to be good parameters.

The most interesting parameter is the number of nodes per hidden layer, as it greatly affects the computational delay. Figure 5.3 shows that the number of nodes affects the accuracy greatly, until about 50 nodes per hidden layer. A MLP with $h_0 = 30$, $h_1 = 20$ and $h_2 = 10$ nodes for the three hidden layers also seems promising, with an accuracy close to the MLP model with 50 nodes per hidden layer. The according confusion matrix can be

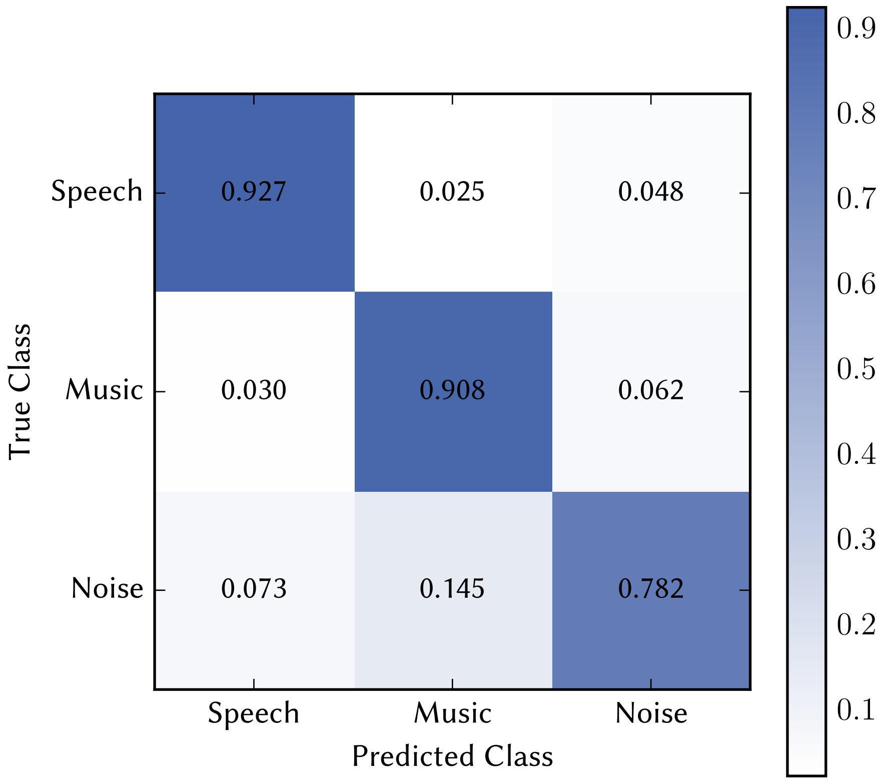

Figure 5.1: Confusion matrix of the accuracy of the MLP tested and trained with the MUSAN data.

seen in Figure 5.1. The computational delay caused by this classification model has been tested to be 1 ms on a single core of an AMD Opteron 6136 Processor with a clock speed of 2.4 GHz.

Parameter	Value
batch size	1000
learning rate	0.001
hidden layer architecture	$h_0 = 30$, $h_1 = 20$, $h_2 = 10$
training steps	$2 \cdot 10^6$
activation function	sigmoid

Table 5.2: Good parameters for a multilayer perceptron given the classification task.

Out of domain test To find out how well the classification model performs on unseen data, the PBC dataset is used as an out of domain test. The out of domain test resulted in an accuracy of 84% compared to the in domain test, which had an accuracy of 87%. The heat map seen in Figure 5.2 shows that the speech precision is quite high with about 90%, and that only 3% of the music is classified wrongly as speech.

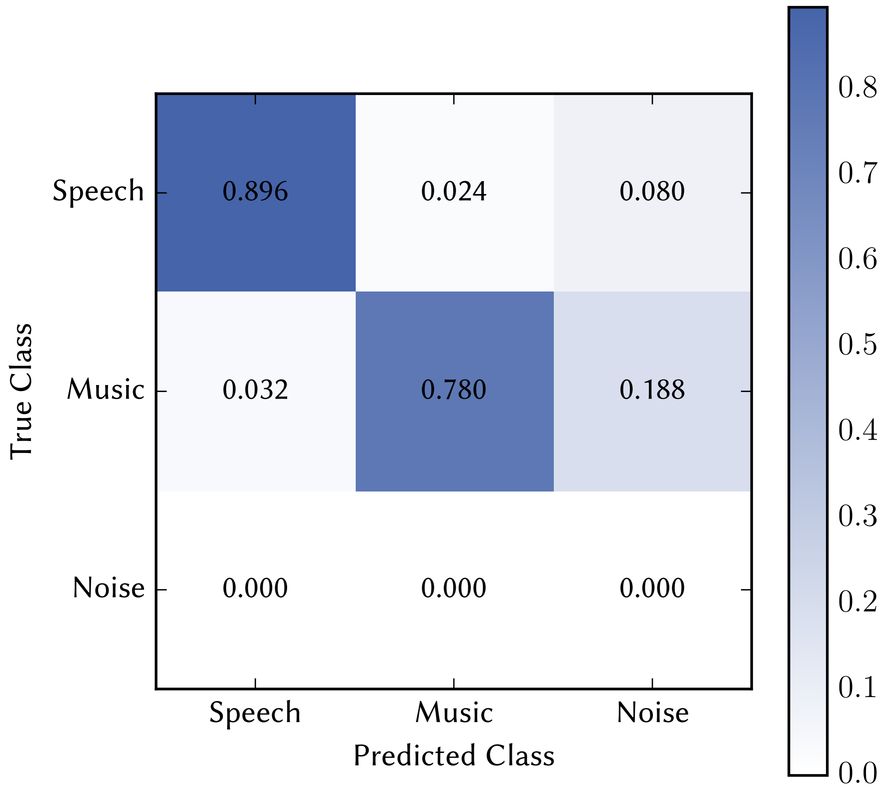

Figure 5.2: Confusion matrix of the accuracy of the MLP tested with the PBC data.

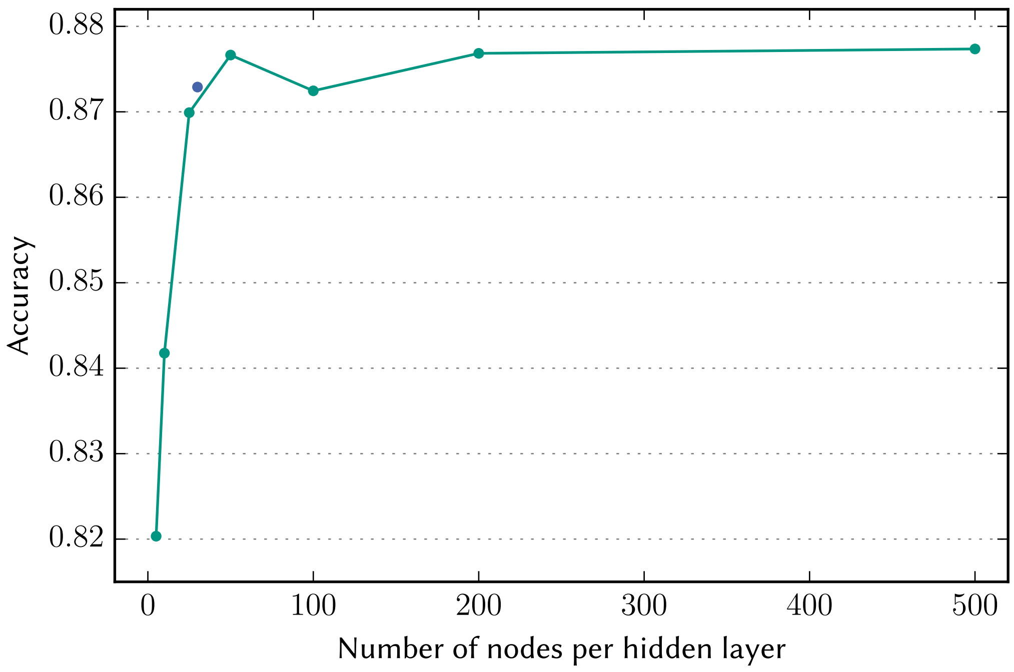

Figure 5.3: Evaluation of the number of nodes per hidden layer. The blue dot represents a MLP with a hidden layer architecture of $h_0 = 30$, $h_1 = 20$, and $h_2 = 10$ nodes.

5.4.2 Support Vector Machine Model

A SVM is used as a comparison classification model to the MLP. The SVM classifies the frames extracted by the feature extraction process with parameters as seen in Table 5.1. A random search with 20 iterations was conducted to find the optimal parameters. The resulting parameters were: A polynomial kernel, a *C* of 0.001, and a gamma of 100. Those parameter led to an accuracy of 83%.

5.4.3 Feature Analysis

To find the most promising feature set multiple tests with different features are performed. The accuracy of the classification is used as evaluation criteria. Additionally the caused inherent delay is used as another evaluation criteria. Tests showed that the feature extraction process with parameter as seen in Table 5.1 causes a computational delay of $<$ 1 ms.

Feature Combinations This paragraph shows the accuracy of purely MFCC based features, compared to features based on MFCC and ZCR. The computational delay is taken into account as evaluation criteria. The results can be seen in Figure 5.4. The advantage of including the zero-crossing is most noticeable for a small delay of $\leq$ 50 ms, possibly because more spectral information becomes more important when less temporal information is given.

Statical Features To decrease the feature space, feature statistics are used as features. Combinations of the statistic functions mean, standard deviation and variance are investigated. The results are shown in Figure 5.5. Using the mean function led to a classification accuracy of 79%, while standard deviation and variance both led to an accuracy of 61%. Combining mean with standard deviation or variance led to an accuracy increase to 86% in both cases. The combination of standard deviation and variance lead to an accuracy of 66%, which is surprising, since the two functions are strongly correlated. Using all three functions lead to a small accuracy increase to 87%.

Mel Frequency Cepstral Coefficients As the number of MFCCs influences the feature space dimension, and therefore the computation cost and delay, a small number of MFCCs is preferred that still provides a high accuracy. As the computational delay is quite small, the accuracy is more important than the feature space dimension. Figure 5.6 shows that more than 20 MFCCs do not increase the accuracy anymore, but less than 20 coefficients decrease the accuracy noticeably.

Inherent Delay A small inherent delay is important for real-time recognition, however the accuracy suffers when there is not enough temporal information. As the inherent delay is directly linked to the frame size and context, those parameters are investigated. Figure 5.7 shows how multiple frame sizes perform given a maximum inherent delay (which fixes the frame context). For a inherent delay below 90 ms a small frame size (10 ms) seems to have a small advantage. Between 100 ms and 180 ms a frame size of 20 ms

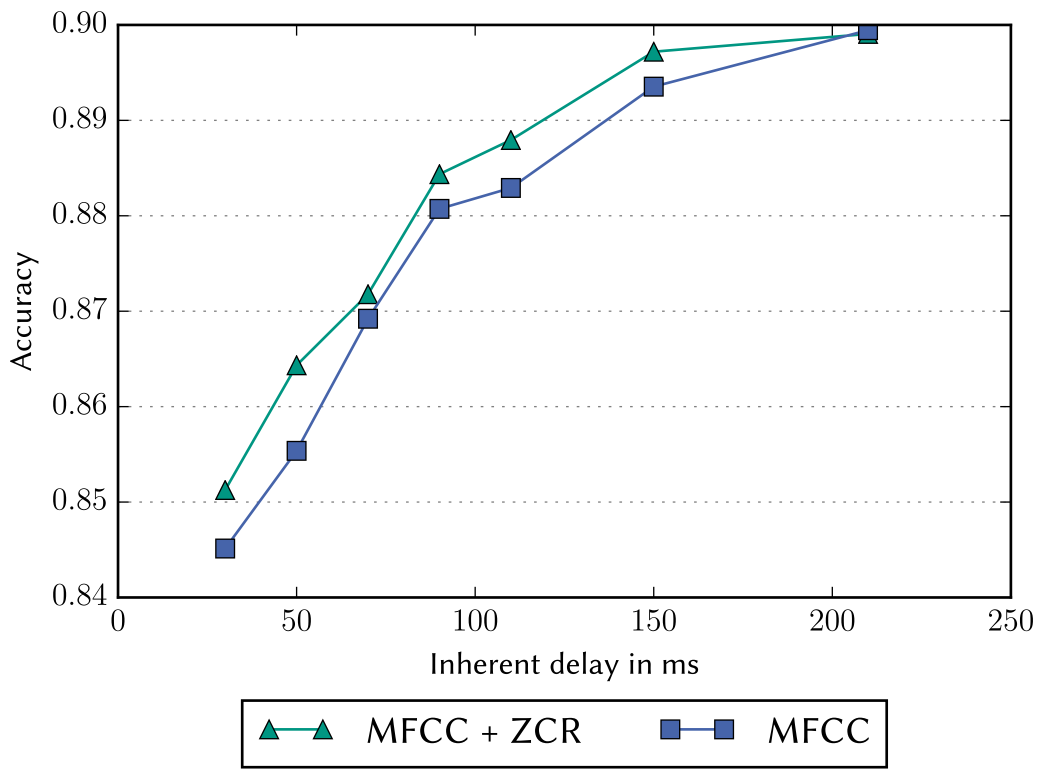

Figure 5.4: Accuracy of MFCC based features compared to MFCC and ZCR based features.

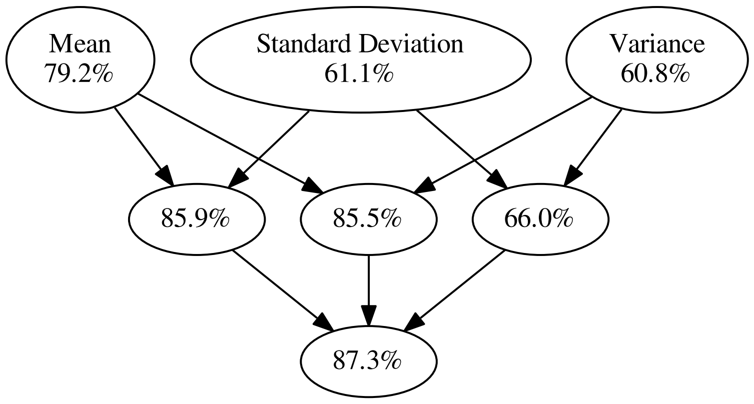

Figure 5.5: The accuracy of the classifier with multiple applied statistical functions and combinations of those. The combinations are indicated by the union of the incoming edges. Using mean, standard deviation and variance led to an accuracy of 87.3%.

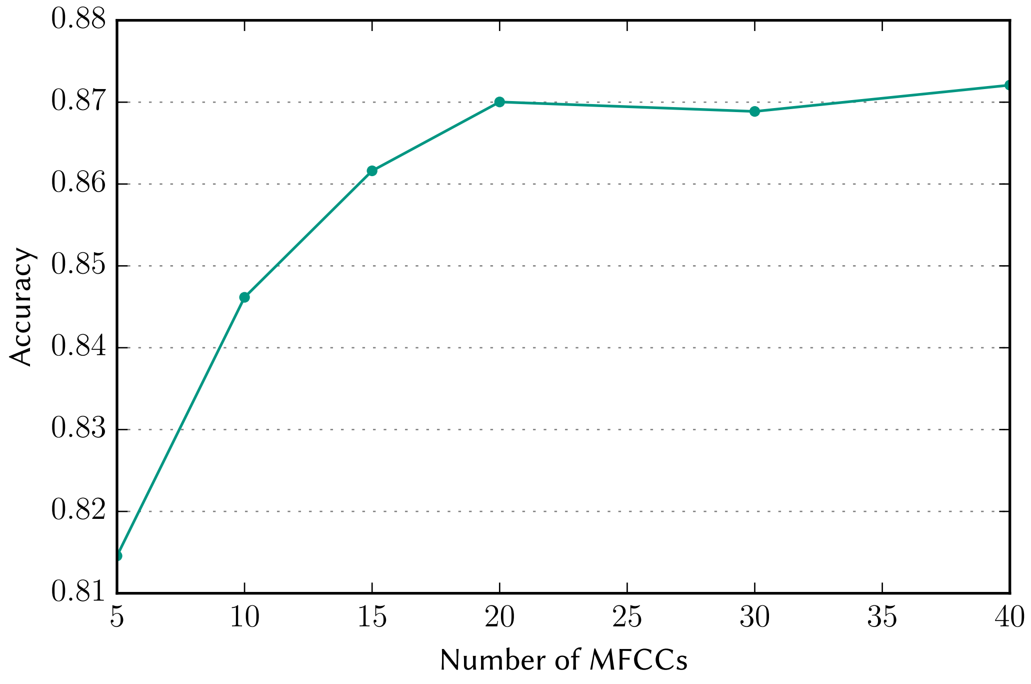

Figure 5.6: Number of Mel frequency cepstral coefficients which are extracted for each feature vector compared to the classification accuracy.

seems to be preferable. Overall there is a clear trend that a bigger inherent delay leads to a noticeable increase in accuracy. A small delay of 30 ms leads to an accuracy of 85%, while a delay of 200 ms reaches an accuracy of 90%.

5.4.4 Smoothing Analysis

The following tests are performed to find out if the smoothing increases the ASR performance, and which smoothing parameters are most successful. An MLP with parameters as seen in Table 5.2 is used as classifier, with features extraction parameters as presented in Table 5.1.

Mode, Erosion, Dilation To evaluate the mode, erosion and dilation context, the inherent delay is split into the relative contribution of each context. The percentage caused by the mode context is denoted as p_m, for erosion as p_e, and p_d for the dilation context. Multiple different relative contributions are investigated. To see if the context contributions should be different when the support method is also applied, an additional test is done for each test file (STC.G and STC.TA) where the support algorithm is also applied, with the parameters $min_{speech} = 2$, $min_{music} = 30$ and $min_{noise} = 30$. The results are shown in Figure 5.8.

The setting $p_m = 0.0, p_e = 0.4, p_d = 0.6$ seems to be good for a inherent delay of 0.2 seconds, but performs worse with higher delays. Probably due to the erosion process, which removes small speech segments. When using the minimum change support making use of all three contexts is always outperformed by a mean context only. This is most

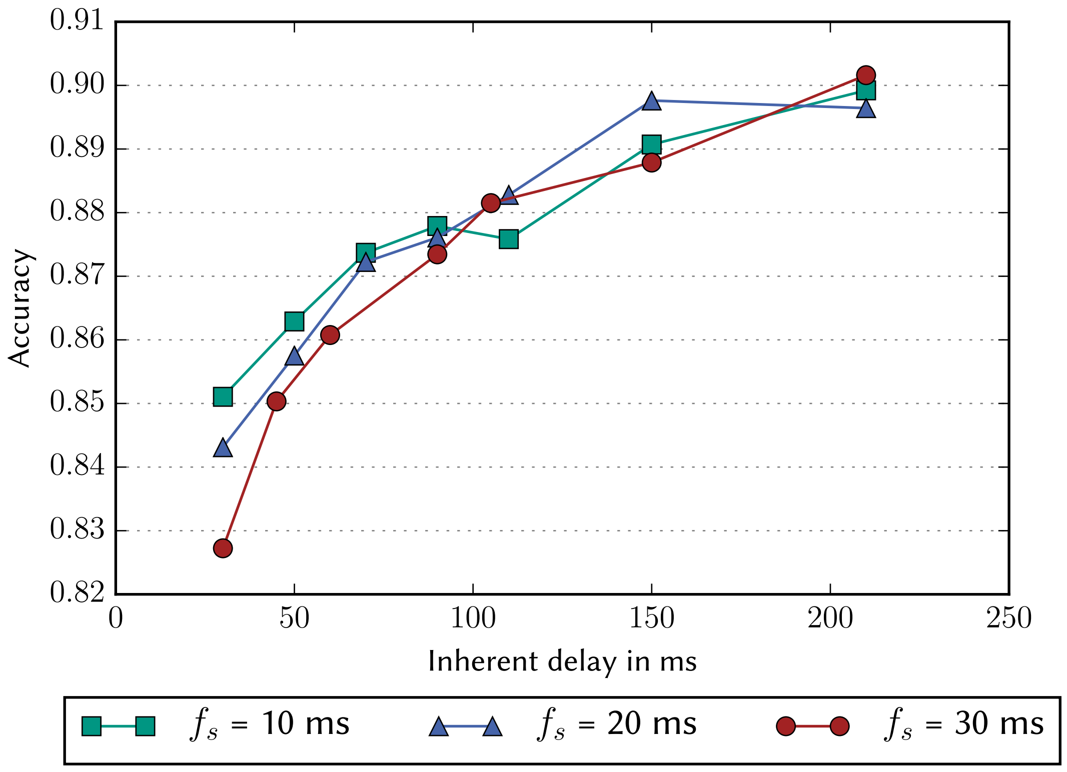

Figure 5.7: Classification accuracy compared to the inherent delay for different frame sizes. The frame context depends on the inherent delay.

likely caused by the fact that for a given inherent delay just using the mean results in the biggest temporal information.

Minimum Change Support As the minimum change support does not affect the inherent delay, much temporal information can be used, however too big support values can lead to too much misclassification and be more harmful for the overall ASR performance. To evaluate the performance of the three parameters min_{music}, min_{speech}, and min_{noise}, a baseline b is used, for which each parameter is calculated as $min_x = b \cdot p_x$, with p_x as a new parameter for each class x. The results can be seen in Figure 5.9.

The parameters $p_{speech} = 0.5, p_{music} = 1.0, p_{noise} = 1.0$ perform well when the music, noise and speech parts are clearly distinct as in test STC.TA, with a baseline smaller than 150. However the parameters perform badly when the distinction between music, noise and speech is not so easy, and the parts change quickly as in STC.G. Reducing p_{speech} to 0.0 or 0.1 yields a slightly worse performance in the STC.TA test, but performs significantly better in the STC.G test. It can be reasoned that a big min_{speech} value omits the beginning of speech segments, which is especially bad with many small segments. Decreasing p_{noise} to 0.2 performs worse on both tests, but especially on STC.G. It seems that multiple small noise segments within speech segments lead to classifying parts of the speech segments as noise (with a small p_{noise}).

$p_{speech} = 0.0, p_{music} = 1.0, p_{noise} = 1.0$ seems to be a good setting of parameters, as it performs well on both tests and both functions have a similar course, indicating robustness against different segment distributions. A baseline of 300 yields the best results, decreasing the WER of STC.G from 29.3% to 24.5%, and the WER of STC.TA from 8.2% to 6.6%.

5.4.5 Segmentation Analysis

This section describes the tests performed to evaluate the affect of the segmentation on the speech recognition task. The parameters used for the smoothing process are shown in Table 5.3. The total inherent delay is 270 ms. The results are shown in Figure 5.10.

All tests that include music (STC.G, STC.TA and STC.P) show a decrease in the word error rate. The tests without music (STC.T and STC.PC) show that the segmenter does not decrease the transcript quality compared to no segmentation when no music is present. The biggest quality increase can be seen in the STC.P test, where a relative WER reduction of 41% could be achieved.

As the STC.TA test is just the STC.T test with added music, the transcript results can be compared. Using the segmentation in the STC.TA test leads to the same WER as the STC.T test, indicating that the segmenter performs as well as removing the music manually. STC.P with segmentation and STC.PC without segmentation show the same results, providing further evidence that the segmenter performs as well as manually removing music.

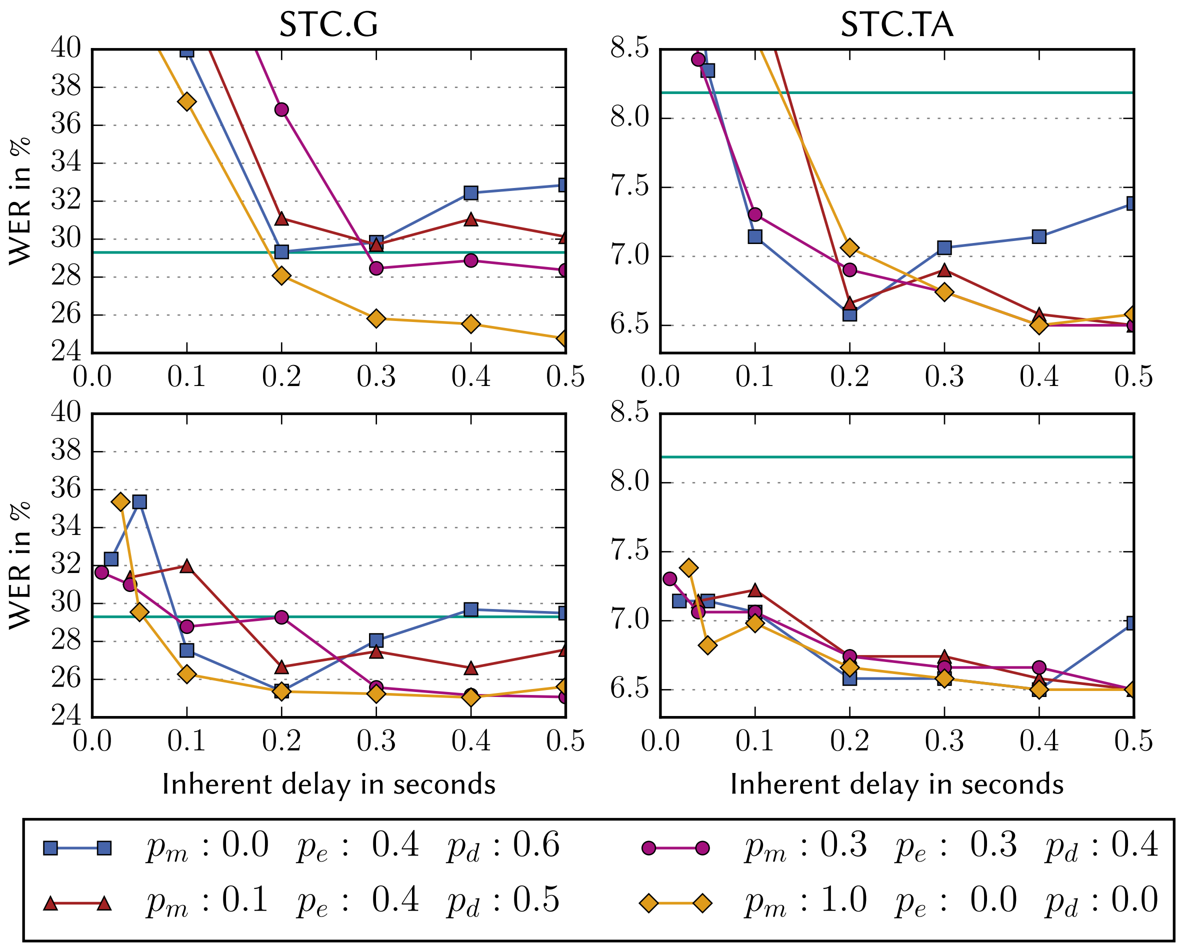

Figure 5.8: This figure shows the speech recognition word error rate for different values of the mean, erosion and dilation context for the STC.G and STC.TA test files. The values are implicitly defined by the relative contribution of each context (p_m, p_e, p_d) to the inherent delay. The first row uses segmentation without the minimum change support, while the second row uses a minimum change support with parameters $min_{speech} = 2$, $min_{music} = 30$ and $min_{noise} = 30$. The green line (unmarked) indicates the WER without any segmentation.

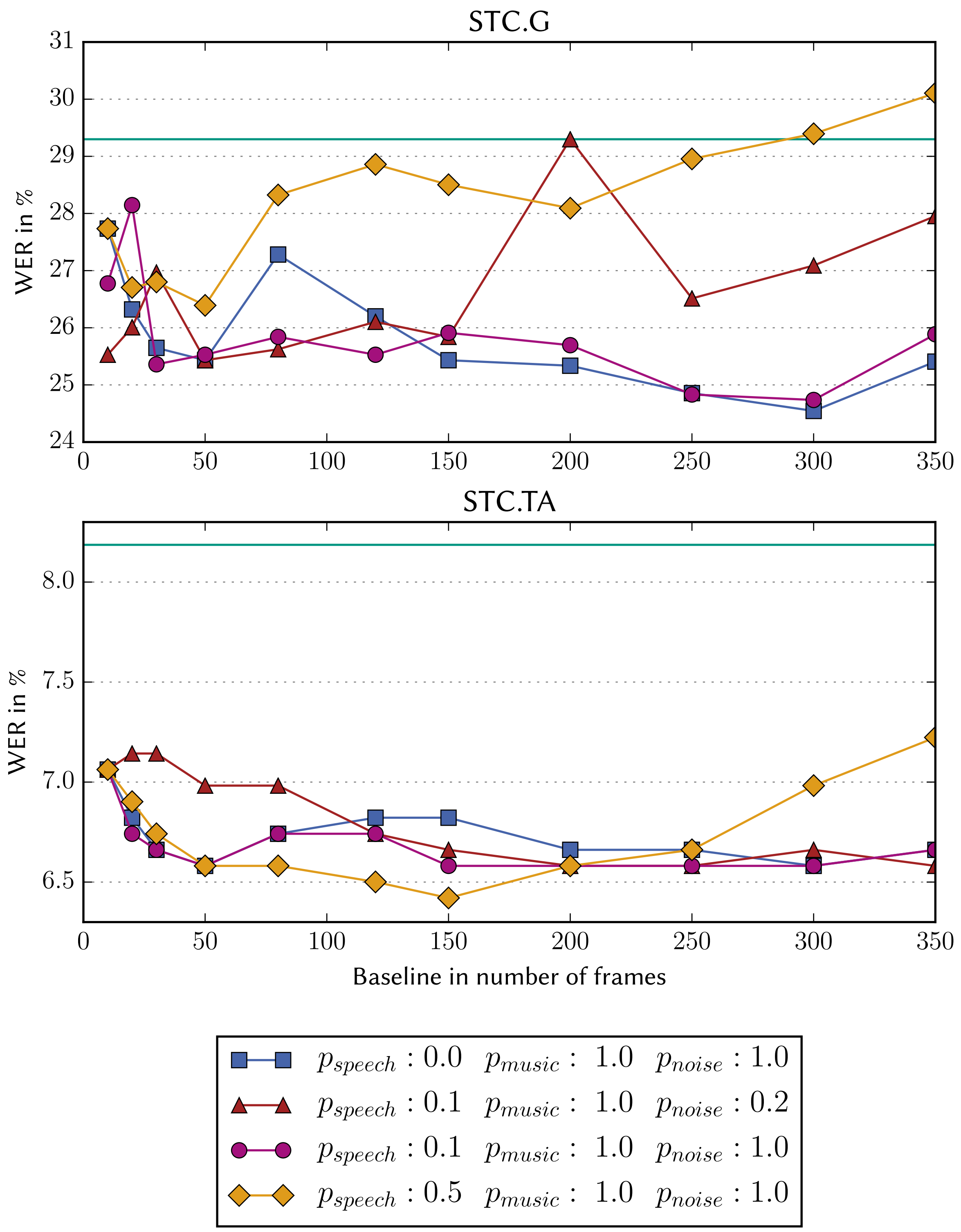

Figure 5.9: Different minimum change support settings are shown for the tests STC.G and STC.TA. A minimum change support for class x is calculated as $min_x = p_x \cdot baseline$. The green line (unmarked) represents the transcription WER without segmentation.

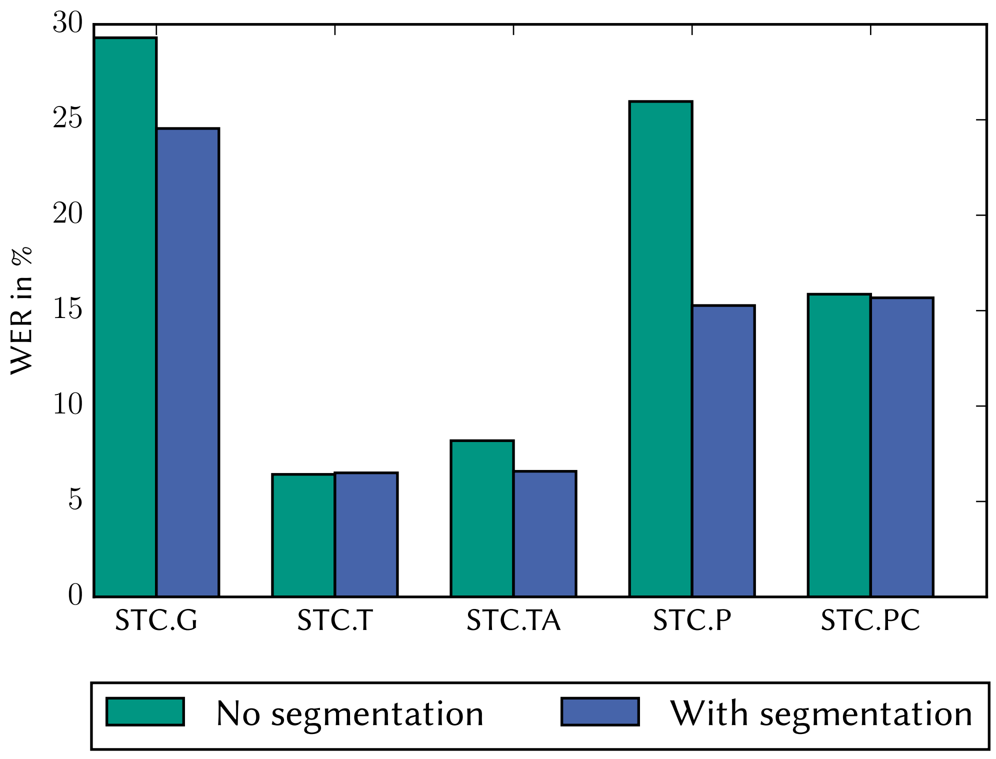

Figure 5.10: The comparison of the transcript quality with segmentation, compared to the quality without segmentation. Measured in the word error rate.

Parameter	Value
mode context	200 ms
dilation context	0 ms
erosion context	0 ms
min_{speech}	0 ms
min_{music}	3000 ms
min_{noise}	3000 ms

Table 5.3: Parameters for the smoothing.

6 Conclusion

In this work a system to increase speech recognition transcript quality by filtering out music and noise segments from the audio stream has been developed and evaluated, while focusing on a low delay to maintain real-time capability. In contrast to previous work, the actual transcript quality has been evaluated. A neural network has been used to classify audio frames into speech, music and noise based on Mel frequency cepstral coefficients and zero-crossing rate features. The feature space has been reduced by taking statistics over multiple frames. The neural network has been compared to a support vector machine, which performed slightly worse. Multiple feature extraction parameters have been optimized with the aim of a good transcript quality while maintaining a small delay. By using the extensive publicly available audio dataset MUSAN the classification results are comparable to possible future work. On this dataset an accuracy of 87 % could be achieved with a real-time delay of 70 ms.

In addition to the classification a smoothing algorithm has been developed, which removes small misclassifications and creates smooth audio segments. The erosion and dilation smoothing step has shown to be unnecessary when the mode smoothing is applied. The minimum change support step, which takes the class of the previous 3 seconds into account, has shown to be important to increase the transcript quality. However test results indicated that when switching to speech the previous seconds should not be taken into account, resulting in a faster switch to speech.

The results have shown that the segmentation algorithm was able to decrease the relative word error rate of a transcript by 41% for an audio stream containing 40 % music, yielding similar results to manually cutting the audio out of the audio stream. Using the algorithm in a real-time environment creates an acceptable delay of 270 ms, plus a negligible computational delay due to its small neural network architecture.

6.1 Future Work

As seen in previous work, different features can be explored to possibly increase the classifier performance. However the results indicate that the smoothing makes up for a classifier that is not perfect and it seems that the task of improving the smoothing is more promising.

One way the smoothing could improved is by increasing the start boundary of the speech segment. The beginning of a speech segment could be shifted by a small amount, to make sure that the start of the speech is included in the segment. This could be achieved by sending the ASR the previous frames when a change to speech is detected by the segmenter. As the ASR would receive a short burst of additional audio it would shortly have a higher delay.

Even though the segmentation algorithm performs as well as a human, a more complex smoothing in addition to a higher classification accuracy could possibly allow for a smaller delay, improving the real-time ability.

Even though the evaluation in this work used an out of domain tests for the audio segmentation, few tests were used and some data is not public. As there is no extensive publicly available test set for audio segmentation beyond pure classification yet, creating such a test set would make work in this research area more comparable.

Bibliography

[Ada10] André Gustavo Adami. “Automatic speech recognition: From the beginning to the Portuguese language”. In: *The International Conference on Computational Processing of Portuguese.* 2010.

[BH03] Michael Berthold and David J. Hand, eds. *Intelligent Data Analysis: An Introduction.* 2nd. Springer-Verlag Berlin Heidelberg, 2003, pp. 169–197.

[BV04] S.P. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004, pp. 215–288.

[BW98] J. S. Boreczky and L. D. Wilcox. “A hidden Markov model framework for video segmentation using audio and image features”. In: *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on.* Vol. 6. 1998, pp. 3741–3744.

[CB04] Ronan Collobert and Samy Bengio. “Links Between Perceptrons, MLPs and SVMs”. In: *Proceedings of the 21st International Conference on Machine Learning.* ACM, 2004, pp. 23–.

[DHS00] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* 2nd. Wiley-Interscience, 2000, pp. 259–264.

[Eri09] Lars Ericsson. “Automatic speech/music discrimination in audio files”. MA thesis. School of Media Technology, Royal Institute of Technology, Sweden, 2009.

[Eyb+13] Florian Eyben, Felix Weninger, Florian Gross, and Björn Schuller. “Recent Developments in openSMILE, the Munich Open-source Multimedia Feature Extractor”. In: *Proceedings of the 21st ACM International Conference on Multimedia.* ACM, 2013, pp. 835–838.

[Goo+13] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay D. Shet. “Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks”. In: *Computing Research Repository* abs/1312.6082 (2013).

[HAH01] X. Huang, A. Acero, and H.W. Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development.* Prentice Hall PTR, 2001.

[HC03] H. Harb and Liming Chen. “Robust speech music discrimination using spectrum’s first order statistics and neural networks”. In: *Proceedings of the 7th International Symposium on Signal Processing and Its Applications.* Vol. 2. 2003, pp. 125–128.

[Hec+13] Michael Heck, Christian Mohr, Sebastian Stüker, Markus Müller, Kevin Kilgour, Jonas Gehring, Quoc Bao Nguyen, Van Huy Nguyen, and Alex Waibel. “Segmentation of telephone speech based on speech and non-speech models”. In: *Speech and Computer*. Springer, 2013, pp. 286–293.

[HL02] Chih-Wei Hsu and Chih-Jen Lin. “A comparison of methods for multiclass support vector machines”. In: *IEEE Transactions on Neural Networks* 13.2 (2002), pp. 415–425.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5 (1989), pp. 359–366.

[IC16] Yoshua Bengio Ian Goodfellow and Aaron Courville. “Deep Learning”. Book in preparation for MIT Press. 2016.

[JM09] D. Jurafsky and J.H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson Prentice Hall, 2009.

[Ked86] B. Kedem. “Spectral analysis and discrimination by zero-crossings”. In: *Proceedings of the IEEE* 74.11 (1986), pp. 1477–1493.

[Las+12] Walter Lasecki, Christopher Miller, Adam Sadilek, Andrew Abumoussa, Donato Borrello, Raja Kushalnagar, and Jeffrey Bigham. “Real-time captioning by groups of non-experts”. In: *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 2012, pp. 23–34.

[Lav+97] Alon Lavie, Alex Waibel, Lori Levin, Michael Finke, Donna Gates, Marsal Gavalda, Torsten Zeppenfeld, and Puming Zhan. “JANUS-III: Speech-to-speech translation in multiple languages”. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. IEEE. 1997, pp. 99–102.

[Log+00] Beth Logan et al. “Mel Frequency Cepstral Coefficients for Music Modeling.” In: *Proceedings of the 1st International Society for Music Information Retrieval*. 2000.

[LZL03] Lie Lu, Hong-Jiang Zhang, and Stan Z Li. “Content-based audio classification and segmentation by using support vector machines”. In: *Multimedia systems* 8.6 (2003), pp. 482–492.

[Mal+00] K. El-Maleh, M. Klein, G. Petrucci, and P. Kabal. “Speech/music discrimination for multimedia applications”. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*. Vol. 6. 2000, pp. 2445–2448.

[Mar+15] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[MB03] Martin F McKinney and Jeroen Breebaart. “Features for audio and music classification.” In: *ISMIR*. Vol. 3. 2003, pp. 151–158.

[Mit97] T.M. Mitchell. *Machine Learning*. McGraw-Hill, 1997, pp. 81–127.

[Ped+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. "Scikit-learn: Machine learning in Python". In: *The Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[PT05] C. Panagiotakis and G. Tziritas. "A speech/music discriminator based on RMS and zero-crossings". In: *IEEE Transactions on Multimedia* 7.1 (2005), pp. 155–166.

[PT14] Aggelos Pikrakis and Sergios Theodoridis. "Speech-music discrimination: A deep learning perspective". In: *Proceedings of the 22nd European Signal Processing Conference*. IEEE. 2014, pp. 616–620.

[Rab89] Lawrence R Rabiner. "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286.

[Ram99] S. A. Ramprashad. "A multimode transform predictive coder (MTPC) for speech and audio". In: *Workshop on Speech Coding Proceedings*. IEEE. 1999, pp. 10–12.

[Rog05] Ivica Rogina. *Spachliche Mensch-Maschine-Kommunikation*. Unpublished, 2005.

[Sau96] John Saunders. "Real-time discrimination of broadcast speech/music". In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*. IEEE. 1996, pp. 993–996.

[SCP15] David Snyder, Guoguo Chen, and Daniel Povey. "MUSAN: A Music, Speech, and Noise Corpus". In: *Computing Research Repository* abs/1510.08484 (2015).

[Sol+01] Hagen Soltau, Florian Metze, Christian Fügen, and Alex Waibel. "A one-pass decoder based on polymorphic linguistic context assignment". In: *Workshop on Automatic Speech Recognition and Understanding*. IEEE. 2001, pp. 214–217.

[WGY03] W. Q. Wang, W. Gao, and D. W. Ying. "A fast and robust speech/music discrimination approach". In: *Proceedings of the Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing and Fourth Pacific Rim Conference on Multimedia*. Vol. 3. 2003, pp. 1325–1329.

[Wik11] Wikipedia. *Support vector machine — Wikipedia, The Free Encyclopedia*. [Online; accessed 30-April-2016]. 2011. URL: `https://en.wikipedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png`.