JACOBS
UNIVERSITY

Echo State Networks Classifier for Automatic Language Identification

by

Valentin Vasiliu

Bachelor Thesis in Computer Science

Prof. Dr. Herbert Jaeger
Name and title of the supervisor

Date of Submission: May 8, 2016

Jacobs University — School of Engineering and Science

Abstract

Automatic Language Identification (LID) refers to the problem of recognizing the language spoken in an utterance sample by a computer system, and it is part of the continuous research efforts in the improvement of human-computer interaction. The complexity of the task rests in the degree of specialization required: from phonologists and linguists creating language models, to sound engineers extracting meaningful signal features from the audio sample and to computer scientists assembling the pieces to obtain a classifier model.

Echo State Networks (ESN) are a variation of the Recurrent Neural Networks (RNN), which preserve the short-term memory property of RNNs without the computational complexity of training such a network, with the condition that certain restrictions are set. Through their practical nature and easy implementation ESNs have attracted considerable interest and have been applied successfully in engineering applications and domains such as communications, pattern generation (e.g. robot modelling, medical analysis), dynamical pattern recognition (e.g. speech recognition), prediction of time series (e.g. stock markets).

This guided research aims to develop an ESN-based classifier for the LID problem. The motivation behind using ESNs rests in their ability to successfully model temporal sequences, i.e. speech sequences for the current task, and consequently provide a fast and efficient alternative approach to other LID proposed solutions.

Contents

1 Introduction

In machine learning many applications employ feed-forward neural structures which are well understood due in part to their non-dynamic nature. Temporal problems can also be solved through feed-forward networks, however generally in such models many parameters are needed and time is not represented in a 'natural' way [1]. A different approach would be using Recurrent Neural Networks which are biologically inspired models, built on top of feed-forward neural networks by adding recurrent network connections between its internal neurons transforming the system into a non-linear dynamical system.

Theoretical results have shown that RNNs can approximate arbitrary finite state automata and dynamical systems with arbitrary precision ("universal approximators") [2]. While RNN might seem capable to model complex temporal tasks, practical difficulties have limited its possible applications, albeit it is worth mentioning that advances in deep learning over the last few years have reduced some of these difficulties. One of the main issues RNN practitioners face is the problem of the "fading gradient" where small changes to the RNN's parameters can lead to non-converging weights [3]. Other issues include, but are not limited to: slow learning, high complexity, hard long-time learning (a possible solution to this would to use a Long Short Term Memory - LSTM architecture, though it doesn't always outperform time delayed neural networks) [1]. Even if the number of neurons in the RNN is reduced to improve the computational efficiency, the expressive power coming from the dynamics of the model is reduced as well.

Reservoir Computing (RC) proposes a different way of understanding and using RNNs, its core idea being to separate the recurrent part (*reservoir*) and the readouts. Echo State Networks and Liquid State Machines are the main initiators of RC, and although their origins are different both of them assume that if the RNN part of the model satisfies certain criteria, supervised updates of all the weights is unnecessary and training the readout should be enough to provide good results for the respective task [1][3].

The properties of echo state networks make them convenient to use in many temporal data applications. Such properties (which will be described in the following sections) also motivated the current research topic: to develop an ESN based classifier for the automatic language identification problem (LID).

Automatic Language Identification (LID as it is commonly known in the literature) is the problem of recognizing the language uttered in a speech sample, by an unknown speaker (i.e. no prior information about the speaker is available). LID is a part of the broader speech recognition area and it has been highly researched over the last few decades and continues to be as new classifier models are developed and new feature selectors are built [4].

There are multiple applications for automatic language identification, for example hotel checking, call redirection in cases where the caller is not a native in the language spoken by the operator (in emergency situations this would be of great importance), and it can serve as input for the speech translation models, which once they know the languages with highest probabilities to match the target speech language they can chose the right model accordingly [4].

The current document is structured as follows. Section 2 describes the research problem and the research objectives in detail, by providing relevant information of the topic from previous literature and an in-depth look into the ESN structure. Section 3 presents the

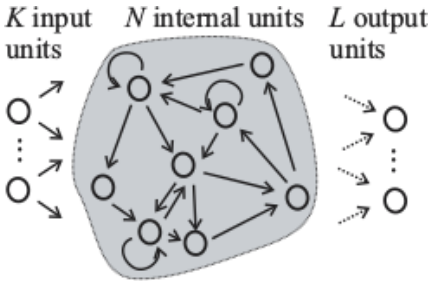

Figure 1: General ESN layout, dashed lines represent trainable connections. Figure taken from [8].

experimental setup as well as the theoretical considerations behind it. Afterwards Section 4 outlines the results of the experiments. The concluding remarks and suggestions for future research are provided in Section 5 and Section 6, respectively.

2 Statement and Motivation of Research

2.1 ESN: General Structure

Echo State Networks (ESN) were developed as a computationally feasible alternative to more general approaches involving Recurrent Neural Networks (RNN) and represent the focus in many research areas and practical applications.

As argued in [2] and [5], ESNs are faster to train and easier to use than other RNNs based structures while successfully being able to model temporal data, some of its applications being described in [1] and [6].

The following section will present the general structure of an ESN as expressed in [5] and [7], explaining the intuition behind its parameter settings, describing the reservoir update equations and the readout training process. The explanations will be focused on gradually deriving a general ESN classifier for temporal sequences. An example of an ESN layout can be seen in Figure 1.

The central part of an ESN is the *reservoir*, a randomly generated, recurrent neural network which primarily serves two functions. The first consists in a non-linear expansion of the input feature sequence into a higher feature space. In a classification task that would imply that if the data input $u(n)$ is not linearly separable in its initial feature space $\mathbb{R}^{N_u}$, they could become linearly separable in the space $\mathbb{R}^{N_x}$ spanned by the states $x(n)$ (i.e. neurons) of the reservoir. Since the weights of the reservoir are not changed during the sampling phase [5] the size of the reservoir can easily be increased with little additional computational costs, however in practice the size of the reservoir is chosen with respect to the complexity of the task. The second function of the reservoir is to provide temporal context, the short term memory capability of ESNs being the reason why they are able to model temporal tasks [5][8].

Both functions should provide a comprehensive signal space in $x(n)$ such that $y^{target}(n)$ (the output layer) could be written as a linear combination of $x(n)$. These are the update equations for a typical ESN reservoir using leaking-integrated neurons [6]:

$$\overline{x}(n) = tanh(\mathbf{W}^{in}[1 : u(n)] + \mathbf{W}x(n-1))$$
$$x(n) = (1-a)x(n-1) + a\overline{x}(n)$$

Here n represents discrete time, $\mathbf{W}^{in} \in \mathbb{R}^{N_x \times (N_u+1)}$ the input connection matrix, $\mathbf{W} \in \mathbb{R}^{N_x \times N_x}$ the reservoir weights, $\overline{x}(n)$ the updated reservoir state's activations and a is the leaking rate, which in essence quantifies the speed at which the dynamics of the reservoir change in time [7].

The update equations could be further changed based on the type of task, teacher forcing could be included where the output is sent as feedback into the reservoir and/or a bias (noise) vector. For the LID task specifically the update equations used preserved the form described above. Other aspects to consider when modelling an ESN are the initial removal of training data due to initial transients, the input connection matrix scaling, and the spectral radius of the reservoir weight matrix to control the dynamics of the system and preserve the echo state property which can be roughly put as fading memory of the input [3]. The general flow of operations in an ESN is provided in Figure 2.

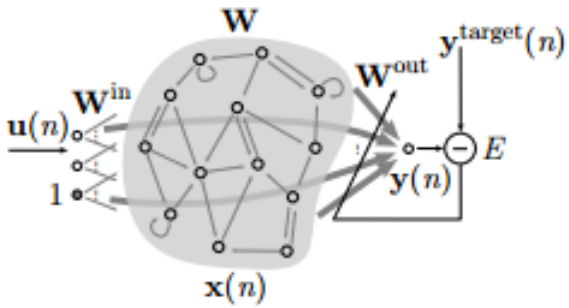

Figure 2: The flow in an ESN model. Figure taken from [7].

The linear readout is given by the equation:

$$y(n) = \mathbf{W}^{out}x(n)$$

Here $y(n) \in \mathbb{R}^{N_y}$ is the activation of the neurons in the output layer, and $x(n)$ the activations of the reservoir neurons. $\mathbf{W}^{out} \in \mathbb{R}^{N_y \times N_x}$ is the output weight matrix trained to optimize the least square error:

$$E(y, y^{target}) = \frac{1}{N_y}\sum_{i=1}^{N_y}\sum_{n=1}^{T}(y_i(n) - y_i^{target}(n))^2$$

T describes the sequence length and N_y the dimension size of the output layer. Since at this moment we deal with a linear partitioning problem, linear regression can be applied, but in order to allow for regularization and avoid numerical instability caused by large $\mathbf{W}^{out}$ values ridge regression [6] is used instead (presented below in matrix form):

$$\mathbf{W}^{out} = Y_{target}^T X(X^T X + \gamma I)^{-1}$$

$X \in \mathbb{R}^{N_c \times N_x}$ represents the collected state activations during the sampling phase where N_c is the number of collected states, I the identity matrix, γ the regularization parameter and $Y_{target} \in \mathbb{R}^{N_c \times N_y}$ is the matrix storing the training data.

In a classification task the output's dimension is equal to the number of classes and $y^{target}(n)$ is created to store 1 in the dimension corresponding to the correct class and 0 for the other classes. The class for a single input sequence $u(n)$ is decided by:

$$class(u(n)) = \underset{k}{\operatorname{argmax}} \left(\frac{1}{\tau} \sum_{n \in \tau} y_k(n) \right) = \underset{k}{\operatorname{argmax}}((\sum \mathbf{y})_k)$$

Where τ is an integration constant (typically the length of the input sequence $u(n)$), and $\sum \mathbf{y}$ is $y(n)$ time-averaged over τ [7]. A different version of this equation will be used for deciding the class of an input sequence for the LID task, and it is further described in Section 3.

2.2 The Language Identification Problem

This section will present a short overview of the Language Identification Problem firstly by providing background context of this area of research, secondly by describing the key characteristics of the problem and lastly by discussing representative approaches in solving this task.

Languages have distinct sound patterns and their detection and comparison are the key in solving the language identification problem. Such cues or characteristics that differentiate between languages are divided in the literature in [4][9][10]:

- cepstral coefficients (based on analyzing the power spectrum of the speech signal)
- acoustic (i.e. phone occurrence frequency per language)
- prosodic (i.e. duration of phones, pitch contours). For example tonal languages like Mandarin and Vietnamese have different pronunciation characteristics than stress languages like English.
- phonotactics are the rules governing the combination of acceptable phones in a language (i.e. legal clusters of phones or phonemes)
- lexical (i.e. the vocabulary of a language)

Individual sounds which are not contained among the phonemes of other languages, like the 'clicks' in sub-Saharan African languages or other unique features can be further integrated [4].

In the next part there will be a short discussion on the features used in speech recognition systems with an emphasis on the mel-frequency cepstral coefficients (MFCC) since the current research method proposed will primarily be using such coefficients as features for the ESN classifier.

Cepstral based features such as MFCC and RASTA-PLP (an acronym for Relative Spectral Transform - Perceptual Linear Prediction) are some of the most known and widely used features in automatic language identification [11]. In essence cepstral based features extract the magnitudes of the frequency bands' energies over the speech spectrum.

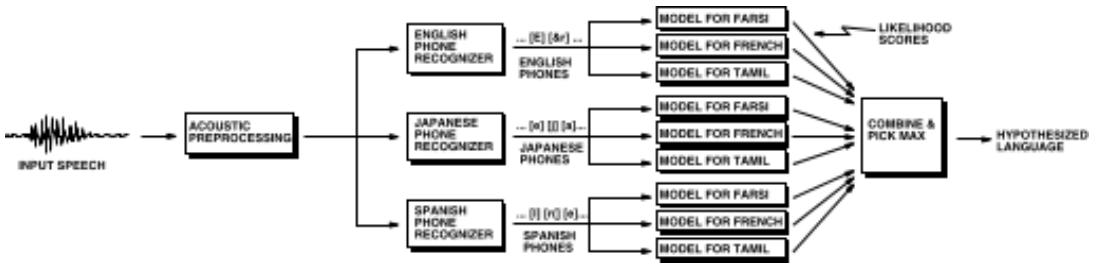

Figure 3: LID system with multiple phone recognizers in parallel. Figure taken from [10].

Previous comparison between MFCC and its alternative has shown little difference in performance when using one or the other, because they practically encode the same information [12]. The current research will focus solely on MFCC as its cepstral features of choice and additional features to be considered will be introduced in later sections.

Several methods have been proposed and applied with success in the automatic language identification problem. A standard approach can be considered the Gaussian Mixture Model (GMM) classification using shifted delta coefficients (SDC) features, with the GMM modelling the acoustic characteristics of a language [9]. In the GMM assumption, each feature vector $\mathbf{v}_t$ at frame t is considered to be randomly sampled from a probability density of multi-variate gaussian densities [11]. A GMM-UBM is a popular variation on the GMM classification, where UBM (Universal Background Model) is a large GMM trained to represent the speaker-independent distribution of features. Concretely, it is aimed for speech selection that relates to the expected speech during the recognition task [13].

Given that languages have different phone collections, there are many LID approaches which focus on phone recognizer systems that hypothesize what phones are uttered in a sample sequence and determine the language by analysing the statistics of the phone sequence [10]. n-gram PRLM (Phone Recognition followed by Language Modelling) type approaches involve tokenizing training messages in each language *l* through a single phone language recognizer and after each symbol sequence is obtained, an n-gram LM(Language Model) probability distribution is generated for each language (typically the sequence information is modelled through HMMs) [11]. From such a language dependent phone recognizer one can construct a multi-lingual phone recognizer by using as frontend preprocessing multiple recognizers [11]. The idea is to run multiple PRLM systems in parallel (P-PRLM) with each single language phone recognition trained on a different language. These are derived either by a mixture of language-dependent and language-independent phones or directly obtained from the training data. An example of such a system is presented in Figure 3.

The bag-of-sounds method differs from the P-PRLM method in that it uses a universal recognizer, which tokenizes the speech sample into a sound symbol sequence, and the obtained symbol sequence is converted into a count vector (bag-of-sounds) [9].

P-PRLM and GMM based approaches involve methods for the estimation of the class-conditional distributions. A different idea consists in applying a discriminative SVM classifier with a Linear Discriminant Sequence kernel that expands into feature space using monomial basis. This approach is described in more detail in [14][15].

Apart from the well-established approaches other methods rely on the usage of Neural Networks with phonotactic and prosodic features [16] and, more recently, due to ad-

vances in acoustic modelling, Deep Neural Networks (DNN) [17]. Current automatic language identification methods [18][19] make use of the popular i-vector based acoustic systems, which represent each utterance as a low-dimensional feature vector [20]. A description of a DNN architecture using i-vector feature vectors and a comparison against other classification methods which are using such features as well is presented in [17].

2.3 Research Objectives

The main objective of the current research is to develop and train an ESN classifier for the language identification problem.

ESNs have been used successfully for pattern generation, for prediction systems and for classification, one of the goals of this research being to test the potential of ESNs in a complex classification task. Examples where they were used for classification purposes include:

- analysis of deterministic dynamics in the motion of real biological beings in a closed environment and their classification based on the identified motion [21],

- time series classification in medical tasks such as predicting dialysis in critically ill patients [22].

Some of the factors which recommend ESN as a classifier are: high dimensional nonlinear transformation of the input feature vectors, short-term memory capability to model sequential data, control of the dynamics of the system via 'metaparameters' (spectral radius, leaking rate, reservoir size and connectivity etc.), and inexpensive training computation when compared to other RNN models, by far more difficult to train and use.

Taking into account the motivation behind using ESN as a classifier for the LID problem, the main objectives of the current research are:

- to develop a robust ESN classifier for the LID problem using MFCC features and to identify 'optimal' parameter settings to improve overall performance,

- to combine MFCC and shifted delta coefficients (SDC) which provide additional information on the rate of change of the MFCC features at time t in the input signal,

- to integrate prosodic features, like pitch and loudness and fuse them together with the MFCC features to obtain a new model,

- comparing the best parameter configurations obtained of the developed classifier models by measuring their accuracy, precision, recall, and Macro F1 scores on differently sized datasets,

- identifying the ESN parameters that contribute the most in improving the performance of the models as well as provide theoretical insights behind their influence.

3 Conducted Experiments

The following section will present each component of the developed models essential to address the research objectives listed in Section 2.3. It will provide theoretical insights and details on the practical implementation for a concrete understanding of the system.

As a short technical overview the ESN classifier toolbox for the current research task is highly modularized, consisting of separate parts for data preprocessing, feature extractors, reservoir construction, readout classification, and efficient multiprocessing capability for testing multiple parameter configurations in order to create a robust environment for analysis and for further functionality that can potentially be added in the future.

3.1 Features

For the current research task 3 main models were developed and analysed, each of them distinguishable by the features used:

- the basic model based solely on MFCC features,
- the 2^{nd} model which integrated shifted delta coefficients in conjunction with using MFCC features,
- the 3^{rd} model which integrated prosodic features such as loudness and pitch on top of the MFCC features.

3.1.1 Mel-frequency cepstral coefficients (MFCC)

An important element in understanding speech is that sounds are produced (or filtered) by the shape of the vocal tract. This shape is encoded in the envelope of the short time power spectrum, and MFCC accurately represent this envelope.

A high level overview on the steps required for MFCC extraction is given below:

- frame the signal into multiple windows,
- for each windowed excerpt compute the periodogram estimate (i.e. the basis modulus-squared of the Discrete Fourier Transform),
- apply the mel filterbank to the power spectra obtained, using overlapping triangular windows, and sum the energies in each filter,
- take the logarithm of the powers of the mel frequencies,
- apply Discrete Cosine Transform on the list of mel log powers.

The MFCC are the amplitudes of the resulting spectrum (in practice the 2-13 coefficients are kept, and the rest discarded) [23][24].

The MFCC features are one of the most popular features employed in LID systems and consequently it was a natural choice in utilizing them for the present task as well. Figure 4 presents an example of the MFCC taken from a speech sample. The audio files were sampled at 44.1 kHz, and divided into 0.025 seconds frames with an overlap of 60% between frames resulting in approximately 1000 frames for a 10 seconds long audio file. The MFCC are extracted from each individual frame resulting in roughly a 1000×13 feature vector per sample [25].

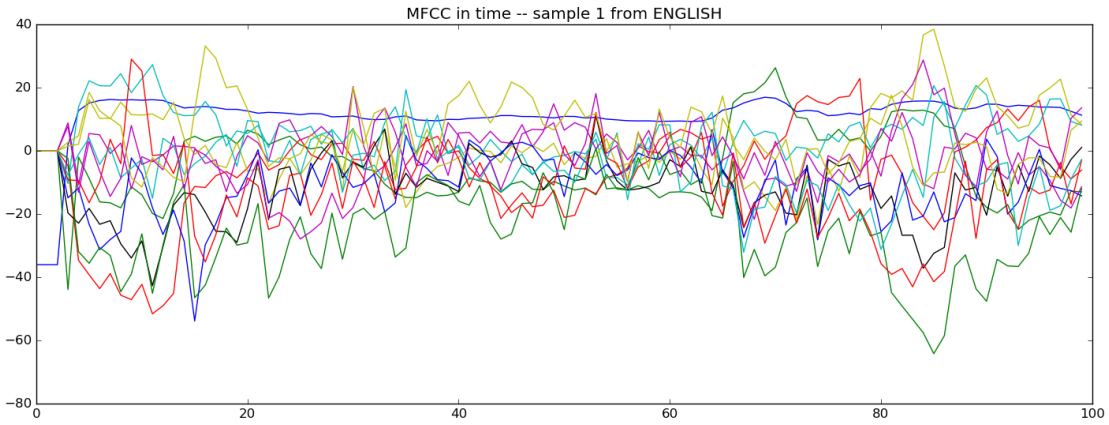

Figure 4: 13 mel-frequency coefficients taken from an English speech sample plotted over the first 100 frames the signal was divided into.

3.1.2 Shifted delta coefficients

To capture long time spectral information from successive frames (i.e. the information speech carries in the dynamics) shifted delta cepstral (SDC) features are added as well [9][26], they are also known as differential and acceleration coefficients, the latter being computed using the differential coefficients. The standard delta (differential) coefficients can be computed as follows [27]:

$$d(t + iP) = c(t + iP + D) - c(t + iP - D)$$

Here $c(t)$ represents the MFCC features at time t and the final feature vector is given by the concatenation of the MFCC features with $d(t + iP)$ for all $0 \leq i < k$. The values for k, P and D are the parameters that need to be set for the SDC.

For the current research a modified version of the SDC was used as documented in [26]:

$$d(t + iP) = \frac{\sum_{d=1}^{D} d[c(t + iP + d) - c(t + iP - d)]}{2 \sum_{d=1}^{D} d^2}$$

The parameter values chosen for computing the modified SDC were $k = 3$, $P = 3$, and respectively $D = 3$ allowing for additionally 39 feature values to the 13 long MFCC feature vector, totalling to a roughly 1000 x 52 feature vector per audio sample. Intuitively the chosen modified SDC values reflect the rate of change of the MFCC features locally and in the near future. The acceleration coefficients (not considered in the current research) are computed in a similar manner only that the delta coefficients are used instead of the 'static' MFCC coefficients.

3.1.3 Pitch and loudness

The model considered in the current section included and fused prosodic features with the cepstral based features. Prosodic features like pitch and intensity have proven to be very effective in both tonal and non-tonal LID, while carrying discriminatory information when compared to cepstral features. Previous approaches which used prosodic features for LID included unsupervised learning pitch contour through GMM or HMM, and subsequent fusion with other features [26]. For the current model the pitch and loudness are extracted from the individual frames the audio signal was split into and added to the MFCC features creating a 1000 $\times$ 15 feature vector per each speech sample.

The loudness is computed through a power calculation using a root-mean-squared operation over the small signal frame (0.025 sec). The value returned after the computation is measured in dB (decibels) and ranges from 0 dB (maximum loudness) to -40 dB (silence) [28].

The pitch detection or the frequency estimation of a signal frame is computed in 2 steps. The first step is computing the Fast Fourier Transform (FFT) of the windowed signal and the second step is using parabolic interpolation to obtain a close estimation of the true frequency [29].

3.2 Network Design

The following section will focus on the ESN component of the system including the parameters used for tuning the network, details on the implementation of the readout training and the method chosen to classify the testing samples.

The constructed networks were continuously improved through intensive tuning and testing. Even if they cannot be considered the most 'optimal' networks in a strict sense of the word they provide excellent results on the dataset used and they showcase the potential of the approach which is the main goal of the current research.

3.2.1 Update equations

The update equations used are the same as mentioned in Section 2.1, for convenience they are repeated below:

$$\overline{x}(n) = tanh(\mathbf{W}^{in}[1 : u(n)] + \mathbf{W}x(n-1))$$
$$x(n) = (1-a)x(n-1) + a\overline{x}(n)$$

Where W^{in} and W are the input connection matrix, and the reservoir matrix, respectively. Both are highly sparse as recommended in the original ESN papers and initialized at the beginning with random values from a $[-0.5, 0.5]$ uniform distribution.

As a result of the experiments conducted the parameters that changed the performance of the models the most were the size of the reservoir and the value of the spectral radius for the reservoir. When analyzing different setups of the parameters it was observed that an increase in the reservoir size contributes to an overall increase in performance and

as the complexity of the model increased by having more features a higher size for the reservoir was beneficial. This is due to the fact that the higher the size of the input signal $u(n)$ is the higher the nonlinear dimensional expansion of $x(n)$ has to be [7]. Even so promising results have been obtained with reservoir sizes not exceeding 500 nodes.

The most optimal value found for the spectral radius across the different models used was around 0.8 – 1.0, though through more extensive testing potentially better values can be found. For the LID problem a higher value for the spectral radius together with a relatively small leaking rate a value implied the need for slow dynamics of $x(n)$ and longer short-term memory in ESN as a consequence of the fast changes in the feature values over time.

3.2.2 Training using Tikhonov regularization

One of the main benefits of using ESNs when compared to other RNN based models is the readout training, which involves training only the output weights ($y(n) = \mathbf{W}^{out}x(n)$) as the reservoir fulfills the echo state property.

In the ESN literature the most popular training method, though other methods to learn linear output weights can be used, is ridge regression or Tikhonov regularization, written in matrix form below:

$$\mathbf{W}^{out} = Y_{target}^{T} X (X^{T} X + \gamma I)^{-1}$$

The X matrix represents the collected reservoir activation states as the feature vectors have been passed through the reservoir. The states are collected only after an initial transient since the network dynamics are influenced in the beginning by the random initialization of the reservoir. As an example, for a feature vector of size 1000 $\times$ 13 the collected activation states assuming an initial transient of 50 frames and a reservoir size of 100 would be of size 950 $\times$ 100.

For a classification task Y_{target} is a collection of y_{target} vectors of size 1 $\times$ number of classes which are 0 everywhere apart from the correct class for the respective state. y_{target} vectors are generated for each of the activation state collected in matrix X. To exemplify if $X \in \mathbb{R}^{N_c \times N_x}$ then $Y_{target} \in \mathbb{R}^{N_c \times N_y}$, where N_c is the number of collected states.

The problem in practice is that N_c can get very large in the order of hundreds of thousands or even millions depending on the size of the dataset and the size of the feature vectors, requiring heavy memory usage.

To avoid such a problem and reduce the memory cost a streamlined Tikhonov regularization has been developed. The main idea is to incrementally compute the $Y_{target}^{T} X$ and $X^{T} X$ matrices by storing only a relatively small number N_b of the states, where N_b is significantly smaller than the total size N_c, but not too small to avoid decreasing the computational speed (in practice N_b was set to 5000). After N_b states have been collected in $X \in \mathbb{R}^{N_b \times N_x}$ and matrix $Y_{target} \in \mathbb{R}^{N_b \times N_y}$ has been generated during the process, partial results are stored in temporary matrices $R = R + X^{T} X$, $T = T + Y_{target}^{T} X$ where R, T are of sizes $N_x \times N_x$ and $N_y \times N_x$, respectively. In the next step another N_b states are

collected and the process is repeated such that at the end the output connection weights can be computed through:

$$\mathbf{W}^{out} = T(R + \gamma I)^{-1}$$

The streamlined Tikhonov regularization presented above has been successfully applied in practice, substantially reducing memory costs with little changes in the computational speed. It allowed parallelizing the process and testing different parameter configurations on multiple processes without a memory bottleneck.

3.2.3 Classification

After the training period of the output weights new samples are run through the network. Considering $u^i(t)$ the input feature values of sample i at frame t, after running it through the reservoir we obtain the output values $y^i(t) = \mathbf{W}^{out}x^i(t)$ which are further collected into an output matrix $Y^i_{out} = [y^i(1)|y^i(2)|\cdots|y^i(n)]^T$, with n being the number of frames for the given test sample. Based on Y^i_{out} the sample is classified according to the following formula:

$$class(u^i) = \arg\max_k \quad count[\arg\max_k(y^i_k(t))|\forall t, 1 \leq t \leq n]$$

Intuitively the formula works in 2 steps. The first step involves finding k, the dimension or class for which $y^i(t)$ provides the highest value and the second step consists in counting the number of occurrences for each k as a result of the first step over the total number of frames. The resulting class k is the class with the highest number of occurrences.

The main benefit of this approach is that it does not require storing the output matrix Y^i_{out}, it only needs to update a frequency vector of the dimensions after each frame processed.

3.3 The dataset

The dataset used for the current research consists of 1200 speech samples, out of which 600 samples denoted as the training dataset are used to pick the best configuration for each model and afterwards the full dataset is used to measure the performance [30].

The samples in the dataset are recorded from 5 different languages: English, French, German, Italian and Spanish, with each language being equally represented in the number of speech samples (i.e. out of the 600 samples in the training dataset each language has 120 samples).

Each audio file is approximately 10 seconds long, in one of the 5 possible mentioned languages involving one unknown speaker.

Different parameter configurations for the ESN models are compared using 5-fold cross validation on the training dataset (i.e. 4/5 of the training dataset samples are used to learn the output connection weights and 1/5 of the samples are used to test the resulting system, the process is repeated 4 more times with the samples being permuted at each

time). The performance of the tested configuration is averaged among the total number of folds.

The configurations of the ESN models that perform the best on the 5-fold cross validation are run in the next step on the full dataset and their performance is evaluated using multiple analysis metrics.

3.4 Performance metrics

To evaluate the effectiveness and reliability of the developed models multiple performance measures have been utilized. Their theoretical value and practical interpretation will be briefly discussed in the current section (for more information refer to [31]):

- *Class Average Accuracy* $= \frac{TP_i+TN_i}{TP_i+TN_i+FP_i+FN_i}$ measures class i effectiveness and it should be close to 1,
- *Class Precision* $= \frac{TP_i}{TP_i+FP_i}$, positive predictive value for class i,
- *Class Recall* $= \frac{TP_i}{TP_i+FN_i}$, also known as true positive rate, measures the effectiveness of identifying true positives of class i,
- *Overall Average Accuracy* $= \frac{1}{L}\sum_{i=1}^{L}\frac{TP_i+TN_i}{TP_i+TN_i+FP_i+FN_i}$, measures the average per-class effectiveness of a classifier,
- *Precision*$_M = \frac{1}{L}\sum_{i=1}^{L}\frac{TP_i}{TP_i+FP_i}$, per-class averaged precision,
- *Recall*$_M = \frac{1}{L}\sum_{i=1}^{L}\frac{TP_i}{TP_i+FN_i}$, per-class averaged recall, measures the effectiveness of a classifier in identifying true positive class labels,
- *F1-score*$_M = \frac{2 * \textit{Precision}_M * \textit{Recall}_M}{\textit{Precision}_M + \textit{Recall}_M}$ is the harmonic mean between *Precision*$_M$ and *Recall*$_M$, and it is desirable to have it as close to 1 as possible.

L represents the number of classes and TP_i, TN_i, FP_i, FN_i are the true positives, true negatives, false positives and false negatives, respectively, for class i. The index M stands for macro-averaging which provides equal weight to every class.

4 Experimental results

4.1 1st model (MFCC)

• 5-fold cross validation on the training dataset consisting of 120 samples for each language

The configuration picked as a representative for the current model was chosen based on the Recall$_M$ score also known as true-positive rate. The main parameters for the configuration include the leaking rate a = 0.2, spectral radius $\rho(\mathbf{W})$ = 1.0, regularization

	English	French	Italian	German	Spanish
Average Accuracy	0.97	1.0	0.99	0.97	0.99
Recall	0.86	0.99	0.96	0.98	1.0
Precision	0.99	0.99	1.0	0.88	0.95

Table 1: Performance metrics on individual classes averaged over all folds

	Overall Avg. Accuracy	Recall$_M$	Precision$_M$	F1-score$_M$
1^{st} model	0.98	0.96	0.96	0.96

Table 2: Performance metrics on the model averaged over all folds

parameter γ = 0.7, and the reservoir size N_x = 250.

- **2-fold cross validation on the full dataset**

In a 2-fold cross validation the dataset is equally split, such that in the current task each class (language) has 120 samples used for training and 120 samples used for testing. In each of the 2 folds the 2 sets are swapped (i.e. the set of samples used for testing in the first fold is the training set in the second fold).

With the model having the parameter settings listed above the following results were obtained:

	English	French	Italian	German	Spanish
Average Accuracy	0.97	0.94	0.95	0.94	0.97
Recall	0.86	0.99	0.83	0.79	0.94
Precision	0.97	0.77	0.92	0.89	0.90

Table 3: Performance metrics on individual classes averaged over the 2 folds

	Overall Avg. Accuracy	Recall$_M$	Precision$_M$	F1-score$_M$
1^{st} model	0.95	0.88	0.89	0.89

Table 4: Performance metrics on the model averaged over the 2 folds

4.2 2^{nd} model (MFCC + SDC)

- **5-fold cross validation on the training dataset consisting of 120 samples for each language**

	English	French	Italian	German	Spanish
Average Accuracy	0.99	1.0	1.0	0.99	1.0
Recall	0.96	1.0	1.0	1.0	1.0
Precision	1.0	1.0	1.0	0.96	1.0

Table 5: Performance metrics on individual classes averaged over all folds

	Overall Avg. Accuracy	Recall$_M$	Precision$_M$	F1-score$_M$
2^{nd} model	1.0	0.99	0.99	0.99

Table 6: Performance metrics on the model averaged over all folds

The configuration picked as a representative for the 2^{nd} model was chosen, as in the previous model, based on the Recall$_M$ score. It is worth mentioning that this model is more complex than the 1^{st} model having a larger feature space and consequently a larger reservoir size was required for better performance.

The main parameters for this configuration include the leaking rate a = 0.1, spectral radius $\rho(\mathbf{W})$ = 0.9, regularization parameter γ = 0.1, and the reservoir size N_x = 400.

As described in Section 3.2.1 the low leaking rate together with a high spectral radius value contributes to a longer short term memory of the input, which is needed due to the fast dynamics in the feature vectors.

- **2-fold cross validation on the full dataset**

With the model having the parameter settings listed above the following results were obtained:

	English	French	Italian	German	Spanish
Average Accuracy	1.0	0.98	0.98	0.99	1.0
Recall	0.99	1.0	0.92	0.96	1.0
Precision	1.0	0.90	1.0	1.0	0.98

Table 7: Performance metrics on individual classes averaged over the 2 folds

	Overall Avg. Accuracy	Recall$_M$	Precision$_M$	F1-score$_M$
2^{nd} model	0.99	0.97	0.98	0.97

Table 8: Performance metrics on the model averaged over the 2 folds

4.3 3^{rd} model (MFCC + pitch + loudness)

- **5-fold cross validation on the training dataset consisting of 120 samples for each language**

	English	French	Italian	German	Spanish
Average Accuracy	0.99	1.0	1.0	0.99	1.0
Recall	0.93	0.99	1.0	0.99	1.0
Precision	1.0	1.0	1.0	0.94	0.98

Table 9: Performance metrics on individual classes averaged over all folds

The main parameters for this configuration include the leaking rate a = 0.2, spectral radius $\rho(\mathbf{W})$ = 1.0, regularization parameter γ = 0.3, and the reservoir size N_x = 300.

	Overall Avg. Accuracy	Recall$_M$	Precision$_M$	F1-score$_M$
3^{rd} model	0.99	0.98	0.99	0.98

Table 10: Performance metrics on the model averaged over all folds

- **2-fold cross validation on the full dataset**

With the model having the parameter settings listed above the following results were obtained:

	English	French	Italian	German	Spanish
Average Accuracy	0.99	0.96	0.98	0.97	0.99
Recall	0.94	0.99	0.91	0.88	0.98
Precision	0.99	0.82	0.99	0.97	0.95

Table 11: Performance metrics on individual classes averaged over the 2 folds

	Overall Avg. Accuracy	Recall$_M$	Precision$_M$	F1-score$_M$
3^{rd} model	0.98	0.94	0.94	0.94

Table 12: Performance metrics on the model averaged over the 2 folds

4.4 Performance summary

	1^{st} model	2^{nd} model	3^{rd} model
Overall Avg. Accuracy	0.98	1.0	0.99
Recall$_M$	0.96	0.99	0.98
Precision$_M$	0.96	0.99	0.99
F1-score$_M$	0.96	0.99	0.98

Table 13: A table summarizing the performance of the 3 models on the 5-fold cross validation for the training dataset

	1^{st} model	2^{nd} model	3^{rd} model
Overall Avg. Accuracy	0.95	0.99	0.98
Recall$_M$	0.88	0.97	0.94
Precision$_M$	0.89	0.98	0.94
F1-score$_M$	0.89	0.97	0.94

Table 14: A table summarizing the performance of the 3 models on the 2-fold cross validation for the full dataset

Through the conducted experiments, it was observed that the 2^{nd} model provided superior results when compared to the other investigated models.

The models were tested on the same configurations and while the experiments are not exhaustive and there is room for further parameter tuning, it was clear that the 2^{nd} model outperformed the other models, with the third one improving considerably on the basic setup.

5 Conclusions

This section will present the conclusions of the current guided research thesis and will restate the main accomplishments achieved.

The automatic language identification problem has been extensively researched over the last few decades, and successful approaches were proposed and continue to be developed as new technologies are created.

The properties of Echo State Networks recommend them as models for temporal sequences such as speech sequences, and based on such properties the main purpose of this research was to create an ESN based classifier for the LID problem.

In the examples found in the literature [21][22] ESNs have provided good results in complex classification tasks, and together with the results of the current research their potential in classification problems is showcased even more.

Many features meant to contain representative information in speech sequences have been proposed, and the literature describing such features and methods employing them is vast. The current research is focused on cepstral based features (i.e. MFCC), shifted delta coefficients obtained from MFCC and prosodic features (i.e. pitch and loudness). These features were successfully utilized in developing efficient and robust models achieving significant results, as presented in detail in Section 4.

6 Further investigation

The following section will conclude the present document and will describe a few ideas worth pursuing for future investigation.

A state-of-the-art feature extraction method, highly popular in current speech recognition tasks, relies on computing the i-vectors (or total variability space approach) acoustic based features [20]. In the i-vector representation approach a sequence of frames for a given utterance is mapped to a low-dimensional feature vector, referred to as total variability space, based on the factor analysis method. The main idea behind factor analysis is that it assumes variability in speaker and channel components, whereas other approaches adapt to speaker-specific characteristics of speech, channel and other subspaces. Driven by the success of Joint Factor Analysis (JFA), different new approaches have been proposed for LID [18][19]. One of the most recent ones involves using Deep Neural Networks as backend classifier and i-vectors for the frontend feature extraction [17]. An objective for future research could be to implement such features in the ESN classifier toolbox, after getting more familiar with the theory behind them and their characteristics.

Another promising contribution could be testing a different classification criterion than the one used in the current research. For example instead of choosing the dimension by counting the most frequent predominant dimension in each frame t the approach stated in [7] which involves picking the resulting class based on the maximal sum computed for each dimension could be used.

Additional suggestions consist of:

- comparing the developed classifier models' performance against other LID classifier results to analyse how the ESN models fair with respect to established LID solutions,
- data preprocessing which can include background noise reduction or amplification, or detecting and removing long silence frames which do not provide meaningful information.

References

[1] Benjamin Schrauwen, David Verstraeten, and Jan Van Campenhout. An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th European Symposium on Artificial Neural Networks*, pages 471–482, 2007.

[2] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148:34, 2001.

[3] Mantas Lukoševičius, Herbert Jaeger, and Benjamin Schrauwen. Reservoir computing trends. *KI-Künstliche Intelligenz*, 26(4):365–371, 2012.

[4] Y.K. Muthusamy, E. Barnard, and R.A. Cole. Reviewing automatic language identification. *Signal Processing Magazine, IEEE*, 11(4):33–41, Oct 1994.

[5] Herbert Jaeger. Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "echo state network" approach. page 48 pp. German National Research Center for Information Technology, 2002.

[6] Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3):335 – 352, 2007.

[7] Mantas Lukoševičius. A practical guide to applying echo state networks. In Genevieve B Orr and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade*, pages 659–686. Springer, 2012.

[8] Herbert Jaeger. *Short term memory in echo state networks*. Number GMD Report 152. GMD-Forschungszentrum Informationstechnik, 2001.

[9] Rong Tong, Bin Ma, Donglai Zhu, Haizhou Li, and Eng Siong Chng. Integrating acoustic, prosodic and phonotactic features for spoken language identification. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 1, May 2006.

[10] Marc A Zissman and Kay M Berkling. Automatic language identification. *Speech Communication*, 35(1–2):115 – 124, 2001.

[11] Marc A Zissman et al. Comparison of four approaches to automatic language identification of telephone speech. *IEEE Transactions on Speech and Audio Processing*, 4(1):31, 1996.

[12] Ben Milner. A comparison of front-end configurations for robust speech recognition. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 1, pages I–797. IEEE, 2002.

[13] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn. Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1):19–41, 2000.

[14] Elliot Singer, Pedro A Torres-Carrasquillo, Terry P Gleason, William M Campbell, and Douglas A Reynolds. Acoustic, phonetic, and discriminative approaches to automatic language identification. In *INTERSPEECH*, 2003.

[15] William M Campbell. A "svm/hmm" system for speaker recognition. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 2, pages II–209. IEEE, 2003.

[16] M Leena, K Srinivasa Rao, and B Yegnanarayana. Neural network classifiers for language identification using phonotactic and prosodic features. In *Intelligent Sensing and Information Processing, 2005. Proceedings of 2005 International Conference on*, pages 404–408. IEEE, 2005.

[17] Ignacio Lopez-Moreno, Jorge Gonzalez-Dominguez, Oldrich Plchot, David Martinez, Joaquin Gonzalez-Rodriguez, and Pablo Moreno. Automatic language identification using deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 5337–5341. IEEE, 2014.

[18] Najim Dehak, Pedro A Torres-Carrasquillo, Douglas A Reynolds, and Reda Dehak. Language recognition via i-vectors and dimensionality reduction. In *INTERSPEECH*, pages 857–860, 2011.

[19] David Martınez, Oldrich Plchot, Lukás Burget, Ondrej Glembek, and Pavel Matejka. Language recognition in ivectors space. *Proceedings of Interspeech, Firenze, Italy*, pages 861–864, 2011.

[20] Najim Dehak, Patrick Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-end factor analysis for speaker verification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(4):788–798, 2011.

[21] A.F. Mitul, M.J.A. Rabin, M. Rakeeb, A. Al Mamun Khan, G.M.S.M. Rana, A. Mollah, and M.H. Rahman. Classification of real time moving object using echo state network. In *Informatics, Electronics Vision (ICIEV), 2013 International Conference on*, pages 1–6, May 2013.

[22] Femke Ongenae, Stijn Van Looy, David Verstraeten, Thierry Verplancke, Dominique Benoit, Filip De Turck, Tom Dhaene, Benjamin Schrauwen, and Johan Decruyenaere. Time series classification for the prediction of dialysis in critically ill patients using echo statenetworks. *Engineering Applications of Artificial Intelligence*, 26(3):984 – 996, 2013.

[23] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357–366, Aug 1980.

[24] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[25] Lyons James. python_speech_features. https://github.com/jameslyons/python_speech_features, 2013. Accessed: 2016-02-08.

[26] Bo Yin, Eliathamby Ambikairajah, and Fang Chen. Combining cepstral and prosodic features in language identification. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 4, pages 254–257. IEEE, 2006.

[27] Pedro A Torres-Carrasquillo, Elliot Singer, Mary A Kohler, Richard J Greene, Douglas A Reynolds, and John R Deller Jr. Approaches to language identification using gaussian mixture models and shifted delta cepstral features. In *INTERSPEECH*, 2002.

[28] Nathan Whitehead. Soundanalyse. http://pydoc.net/Python/SoundAnalyse/0.1.1/, 2008. Accessed: 2016-02-15.

[29] endolith. Frequency estimation methods in python. https://gist.github.com/endolith/255291, 2009. Accessed: 2016-02-15.

[30] TopCoder. Spoken languages. https://community.topcoder.com/longcontest/?module=ViewProblemStatement&rd=16498&pm=13845, 2015. Accessed: 2016-02-08.

[31] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.