Spoken Digit Recognition with Time-delay Reservoir: tutorial.

1 Outline of the task

Main goal is the identification of one specific digit out of other spoken digits.

Dataset is a subset of the TI46 speech corpus and consist of ten digits, from zero to nine (or one to ten), each uttered ten times by five different female speakers, resulting in 500 speech fragments sampled at $12.5\,kHz$.

TI46 – The Texas Instruments 46-Word Speaker-Dependent Isolated Word Corpus (TI46) is a database of speech which was originally designed and collected at Texas Instruments, Inc. (TI) in 1980, and used initially in performance assessment tests of isolated-word speaker-dependent technology. `http://www.ldc.upenn.edu/Catalog/readme_files/ti46.readme.html`

The TI46 speech material was recorded in a low noise sound isolation booth, using an Electro-Voice RE-16 cardoid dynamic microphone, positioned two inches from the speaker's mouth and out of the breath stream.

Up to now we have only pre-processed by Lyon Passive Ear model (see below) digit samples from TI46 dataset.

2 Ear cochlear models

Skipping many biological detail, a human ear sound processor is schematically presented in Fig.1.The ear is made up of three parts, and sound for a person who has normal hearing passes through all three on the way to the brain. The outer ear is made up of the outer, visible part of the ear and the ear canal.

When a person is exposed to a sound, the outer ear captures the sound vibration and sends it through the ear canal to the middle ear, which consists of the eardrum and three tiny bones. The sound vibration then causes motion in the three tiny bones, which makes the fluid in the cochlea move. The motion of the fluid stimulates the hair cells, which are thousands of tiny hearing receptors inside the cochlea. The hair cells bend back and forth and send electrical signals to the hearing nerve, and the hearing nerve then carries these signals to the brain, where they are interpreted.

It is common practice for speech recognition applications to preprocess the sound in order to enhance the speech-specific features to facilitate the recognition process. Many different preprocessing techniques exist. Several names are listed below:

1. MFCC (Mel-Frequency Cepstral Coefficients)
2. Lyon Passive Ear
3. Seheff cochlear

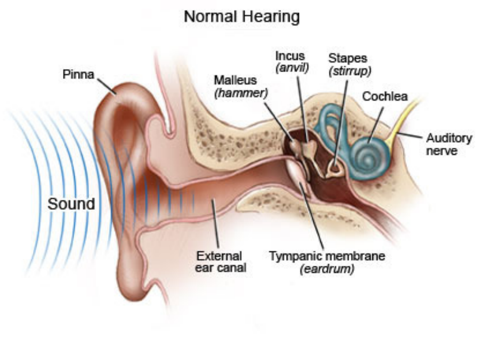

Fig. 1: Structural image of an ear.

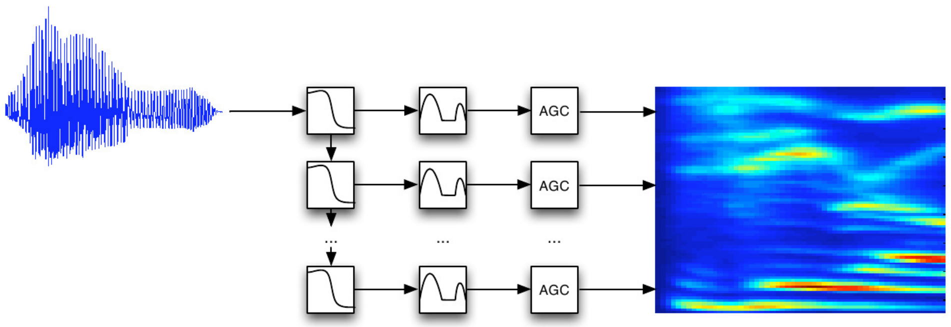

Fig. 2: Schematic overview of the elements of the Lyon cochlear model: a cascading filter bank followed by half-wave rectifiers and adaptive gain controllers (AGC).

4. Patterson-Holdsworth-Meddis

Here we will mention only two of them: most common MFCC and Lyon Passive Ear which is used below in this document.

Mel-Frequency Cepstral Coefficients (MFCC) is a very popular - if not the de facto standard - preprocessing technique in the field of speech recognition. The MFCC are calculated as follows: (1) the sample data is windowed using a hamming window and a FFT is computed, (2) the magnitude is run through a so-called mel-scale filter bank and the $\log_{10}$ of these values is computed, (3) a cosine transform is applied to reduce the dimensionality and to enhance the speech-specific features of the input. The result is the so-called cepstrum.

The Lyon Passive Ear model is a model of the human inner ear or cochlea, which describes the way the human cochlea transforms a sound into a series of neural spikes generated by the hair cells present inside the inner ear. The model consists of a filter bank which closely resembles the selectivity of the human ear to certain frequencies, followed by a series of half-wave rectifiers and adaptive gain controllers both modeling the hair cell response. This form of preprocessing is computationally more intensive than the MFCC front end. It takes about three to five times as long to compute on a conventional processor. Let us consider Lyon model in more details. Schematic overview of the elements of the model is presented in Fig.2. Number of frequency channels can be varied. MATLAB toolbox which performs Lyon Passive Ear transformation (and more) has name 'Auditory Toolbox' and can be found here `http://cobweb.ecn.purdue.edu/~malcolm/interval/1998-010/`. In presented toolbox number of frequency channels can be calculated automatically based on sample rate of initial sound signal. For example, digit samples from TI46 dataset have sample rate equal to 12.5 kHz and after transformation its cochleagrams have 86 channels. The simple example of usage of the toolbox can be written as follows:

```matlab
% Load wav file
% 'fs' is a sample rate
% 'bits' is number of bits
[privet, fs, bits] = wavread('privet.wav');

% generate time scale from sample rate 'fs' and length of 'privet'
t=linspace(0,(1/fs)*length(privet),length(privet));

% Plot original sound time-trace
figure(1)
plot(t,privet)

% Calculate cochleagram: 2D matrix with frequency channals as columns
% and time samples as rows;
% number of frequency channals is defined by sample rate 'fs';
% number of time samples is defined by time window size 'win': initial time
% trace is devided on samples with 'win' length and processed by Lyon model.
win=120;
cochleagram=LyonPassiveEar(privet,fs,win);

% Plot cochleagram as 2D image
figure(2)
imagesc(cochleagram);
```

The result of application of Lyon model to the manually recorded wav file with word 'Privet!' is shown in Fig.3. The word was recorded with sample rate 12 kHz. The corresponding cochleagram has 86 channels

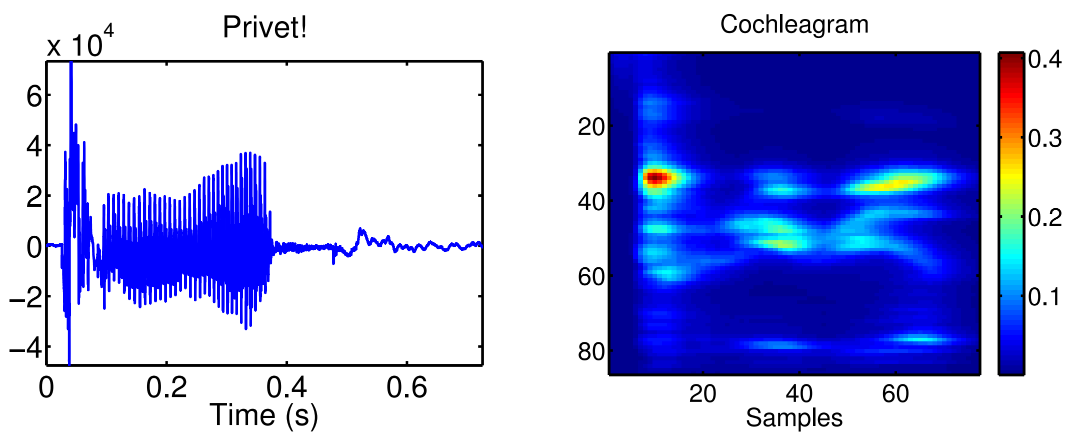

Fig. 3: Time trace and cochleagram of manually recorded word.

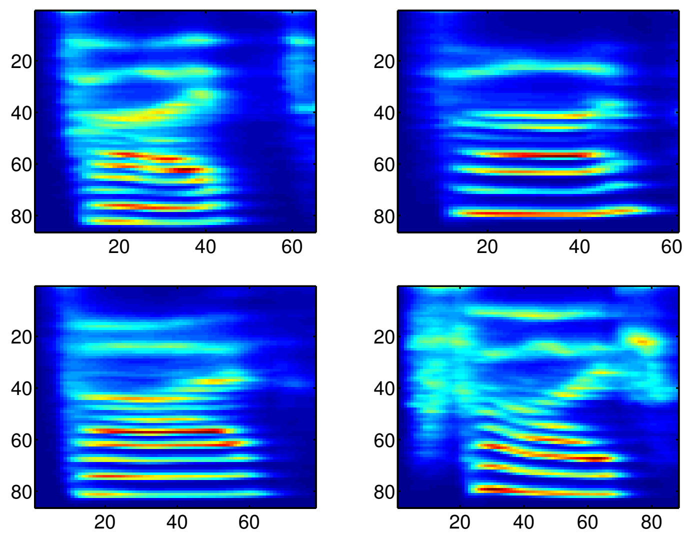

Fig. 4: Four examples of same word ’six’ uttered by different persons.

and 77 time samples total.

3 Dataset examples

Pictures in Fig.4 are presenting several examples of cochleagrams from TI46 dataset. It is clear seen that same digit but pronounced by different persons are close to each others. Moreover one can notice (see Fig.5) that two phonetically closed digits (two and ten) produce quite similar cochleagrams. Thus it may be concluded that cochleagrams generated by Lyon ear model enhance speech accented features of spoken material.

4 Mathematical model

The mathematical model of presented above experimental setup can be written in terms of time-delay differential equations. It is an electro-optical delay system with one or multiple delay feedback loops. The system can be modeled by a delay-differential equation,

$$\tau \frac{\mathrm{d}x}{\mathrm{d}t} + x(t) = \beta \sin^2 \left[x(t - \tau_D) + \gamma v(t) I(t) + \phi_0 \right], \tag{1}$$

where total delay time is denoted by τ_D.

This model can be used to simulate the reservoir which can help in finding appropriate set of parameters for further experiments. However this tutorial is based on real experimental material. Further simulation works are needed to get good agreement of theoretical investigation and experiments. Nevertheless some preliminary simulation were done.

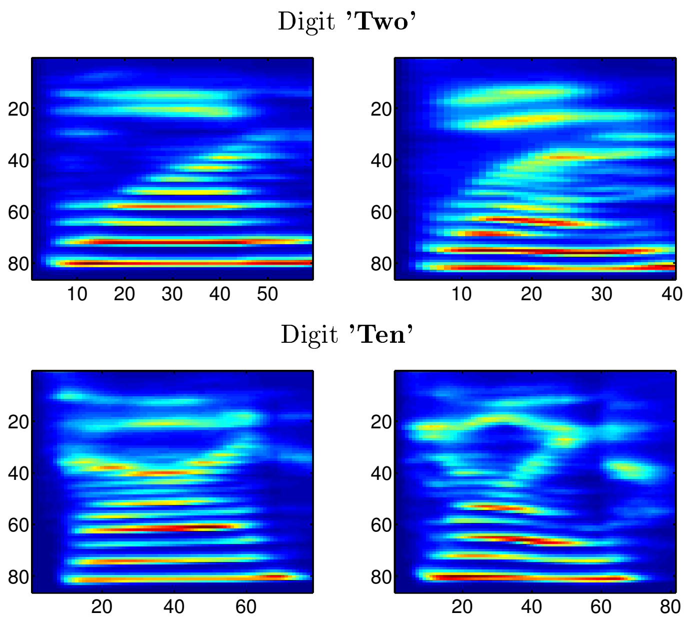

Fig. 5: Examples of cochleagrams for digits 'two' and 'ten' uttered by different persons.

The part of the code with simulate reservoir is shown below.

```
neurwidth=0.2;
%T=nrofneurons*neurwidth;
beta=0.7;
gamma=0.5;
Phi0=pi/4;

%% Input
inkr=5;
binwidth=maskbinwidth*inkr;
h=neurwidth/binwidth;
in=extend(in,binwidth);
in_trans=[zeros(1,floor(500/h)) in];
nsteps=length(in_trans)+nrofneurons*binwidth;
h=neurwidth/binwidth;

%% RC iterations
x=zeros(1,nsteps);
y=zeros(1,nsteps);

% History
x(1:nrofneurons*binwidth)=beta*sin(Phi0)^2*ones(1,nrofneurons*binwidth);

for i=nrofneurons*binwidth:nsteps-1,
    xd=x(i-nrofneurons*binwidth+1)+gamma*in_trans(i-nrofneurons*binwidth+1)+Phi0;
    x(i+1)=x(i)+h*(-x(i)+beta.*sin(xd).^2);
    y(i)=xd;
end

xout=x(end-length(in)+1:inkr:end); %-x(end-length(in));
```

The simulation uses the simple Euler integration method to produce time-trace of reservoir response on input signal. After that the time-trace is down-sampled and cut on equal to number of neurons pieces and stacked in reservoir state matrix. This matrix is used for cross-validation instead the matrix obtained in experiment. The preliminary results for simulation are close to the result from experiments and variate in range from $WER = 0.04$ to $WER = 0.00$ depending on mask and cross-validation partition.

5 Experimental setup

The experimental system is an electro-optical delay nonlinear system with is capable to produce a wide range of possible regimes from steady state and simple periodic motions to fully developed chaotic behavior. By its nature the system is infinite-dimensional. There are several control parameters of the system

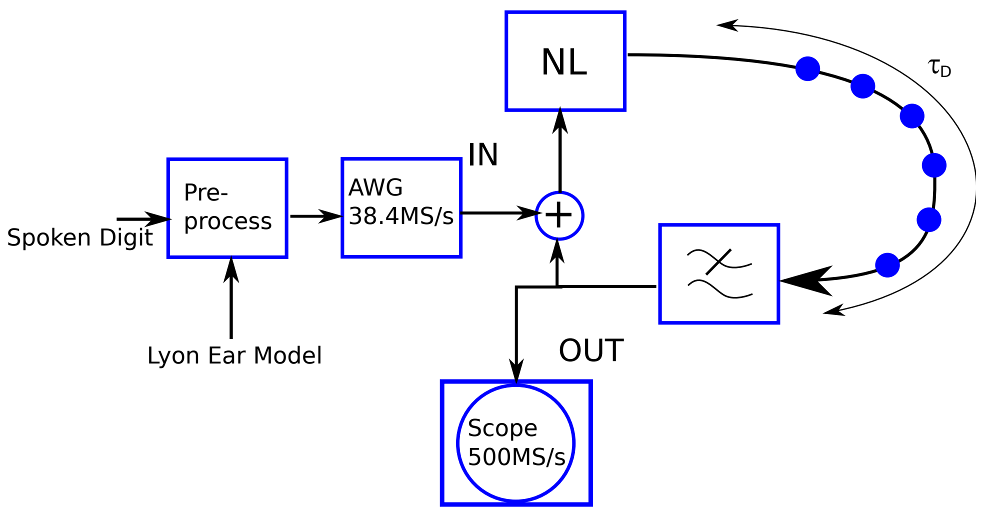

Fig. 6: Schematic representation of experimental setup.

which change its dynamics. The most important are nonlinearity β, delay time τ_D, response time τ and phase operating point ϕ_0. It is believed that the most effective operating point of LSM is a point on the boundary of the stability of a steady state. This means that in absence of input signal the system is in a stable steady state, but it quickly might leave this state when an input is switched on. The input impulse pushes the system to the transient dynamical regime: it evolves toward a steady state but might be interrupted by next input pulse. The characteristic time for transient behavior is defined by transient time τ.

Technical parameters of the setup can be listed as follows,

- number of neurons is equal to 400. Neuron width is equal to 0.2τ, where τ is the main relaxation time of the system (see mathematical model above);
- total delay time $\tau_D = 20.58\,\mu s$. Total delay time should cover all 400 neurons with width 0.2τ. It implies that relaxation time τ and delay time τ_D are coupled by relation: $\tau_D = 400 \times 0.2\tau$;
- input sampling rate F_s produced by AWG: $38.369\,MS/s$. The input sampling rate is calculated in such way that one sample of cochleagram (column) fits the delay time τ_D. Before insertion to the reservoir as an input signal, one sample of cochleagram should be extended by mask on 800 (two identical values for each neuron to maintain the square shape adopted for each sample; it is equivalent to 'sample and hold' procedure for a waveform generators) samples (for detail see Sec. *Input signal design*). This directly implies the relation: $F_s = 800/\tau_D$;
- output sampling rate for the response recorded by Scope: $500\,MS/s$. Output recording frequency is oversampled (by factor ~ 10 in comparison to input sampling). This increases the degree of freedom in choosing the exact values (temporal position) in the reservoir states formation.

6 Input signal design

One of the most important element of the system is the input signal. In the experimental setup it is denoted as IN and in mathematical model as $I(t)$. To be more rigorous the input signal $In(t)$ is formed by the information input $I(t)$ modulated by a mask $v(t)$ and multiple by amplitude γ: $In(t) = \gamma v(t) I(t)$. The mask is a periodic function with period τ_D, total delay time. In the case of speech recognition task, the mask is a sparse 2D matrix with some *sparse density* (typical value is less then 0.1) which determines connections between input sample and neurons of the reservoir: this connection is usually performed through an injection of the input signal into various spatially distributed nodes of a neural network which is forming the reservoir. In the case of a delay dynamics reservoir, this spatial distribution is replaced by a temporal one, spreading the input signal over the time delay, which is viewed as a pseudo spatial space. The sparse density of a matrix is the number of nonzero elements divided by the total number of matrix elements. Nonzero elements of matrix is normally distributed with 0 mean and variance 1. The detailed steps to produce the input signal from sound digit is demonstrated below.

Fig.7 demonstrates basic elements of the input signal design including Lyon ear model transformation and masking modulation. The Lyon passive ear model which transforms the input sound signal $d(t)$ into

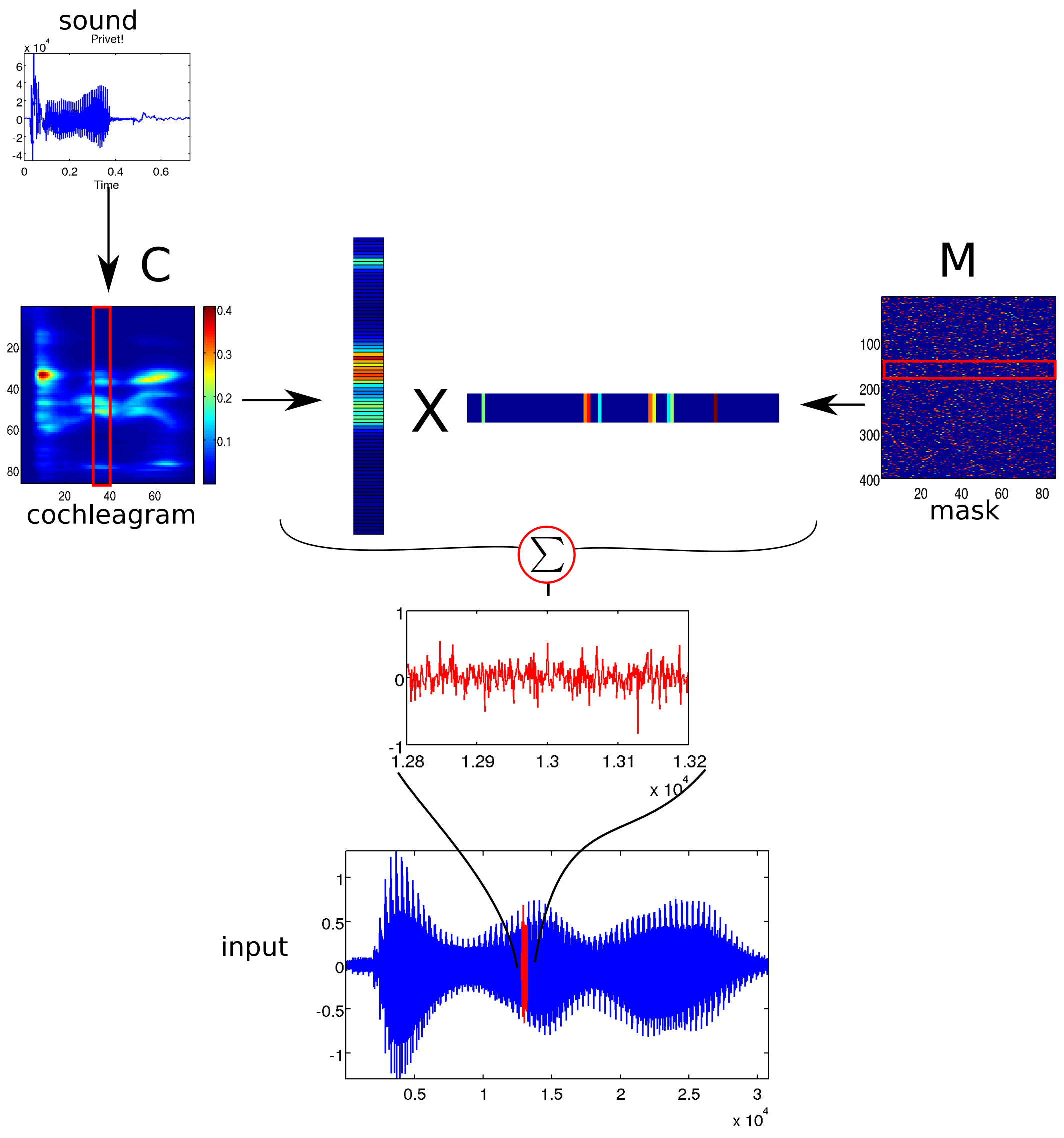

Fig. 7: Input signal design overview.

time-frequency 2D matrix $C[n \times 86]$ (where n is width of a digit in terms of 2D cochleagram, typical values for n are in range from 50 to 100) is more or less clear. For details one can check the MATLAB code `LyonPassiveEar.m` from the Auditory Toolbox mentioned above. After this transformation the input signal is represented by a cochleagram C with 86 frequency channels (spanning along the vertical axis in Fig.2–7), and with a number of samples in time which depends on the spoken word (in the range of 50–100 along the horizontal axis of the same Fig.). In this case an input sample $c_i = \{c_{i,j}\}_{j=1...86}$ is column with 86 values. The mask $M[86 \times 400]$ is a sparse matrix (with many 0's) of random real number between -1 and 1. To produce 400 samples (each for single neuron) from one column sample c_i one should multiply it by each row (element-wise) of the matrix M and sum,

$$In_{k,i} = \sum_{j=1}^{86} c_{i,j} \times m_{j,k}, \quad k = 1...400.$$

where i is an index of time delay and k is an index of sample within a time delay. It produces 1D signal $\{In_{k,i}\}_{k=1...400}$ with 400 numbers for each sample c_i. Total number of elements of input signal for one specific digit is equal to $400 \times n$, where n is the number of temporal samples after Lyon model transformation. To make the situation more clear let's make simple calculation on number of input samples needed for one spoken digit to be inserted in the reservoir. Typical size of samples after Lyon model transformation ranges from 50 to 100. Let for example $n = 70$. Number of samples is equal to $2 \times 400 \times n = 56000$, where 2 is special multiplier to extend input signal in such way that each unique sample appears consecutively twice thus forming a 'sample and hold' square waveform for each value of In. It gives the reservoir to adopt this sample value and spread it time between neurons.

7 LSM structure: inputs, reservoir and targets

Liquid State Machine analysis of experimentally obtained results were made in framework of the MATLAB `Reservoir Toolbox` designed by Ghent University `http://reslab.elis.ugent.be/rctoolbox`. The state of reservoir in time t is the state of all nodes of the system. The state of reservoir can be described by the expression,

$$\{x(t - k \cdot \delta\tau_D)\}_{k=1...N_{nod}} \tag{2}$$

In other words, the state of reservoir is the set of time-traces for all nodes. In simulation it is a matrix with many columns. It is obvious that reservoir states renew all its elements after total time delay τ_D. It means that to collect all information about reservoir dynamics it is enough to record reservoir states separated by τ_D. Fig. 8 shows the process of formation of a reservoir state. Fig. 9 shows basic structure of LSM processing. First column is initial inputs which activate reservoir. Reservoir time-dependent reaction is recorded in states of all neurons separated by total delay time τ_D. The result of recording is shown in second column. The targets (or true answer in term of speech recognition) are presented in corresponding matrices extended in time with zeros everywhere and ones in correct digit position.

8 Cross-validation: training, regression, testing.

To validate the performance of constructed reservoir all spoken digits of the total dataset go through cross-validation process. The hole set of digits is divided on n_p equal and non-intersecting subsets randomly, where n_p is integer number which can be called as *partition size*. For illustration see Fig.10, where $n_p = 10$. One subset can be chosen as a test subset and at the same time all other subsets should be chosen as the train pool. After training and evaluation one can collect errors and continue cross-validation procedure with the other arrangement of the partition. Next step is to take another subset (from n_p previously generated) as the test subset and the rest of the partition as the train pool. This procedure should be done for every subset from n_p subsets to be chosen as test one. It guarantees that every digit will be chosen only once as test digit.

Training process contains several common steps. First one should collect all train digits samples and corresponding reservoir states a into big matrix A. All targets for train subset form target matrix B which goes through *Fisher labeling* procedure to increase separability of digits based on statistics of appearance. Main step of training is to find best read-out matrix W in such way that,

$$A \times W \simeq B.$$

To find optimal approximation for the matrix W there are several methods. The simplest one is the linear regression. It can be written in following way,

$$W_{opt} = \arg\min_{W} \|AW - B\|^2 .$$

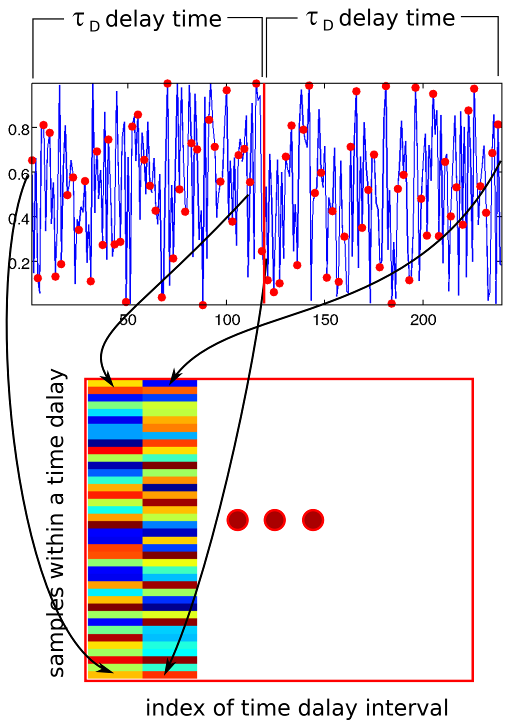

Fig. 8: Reservoir states collection procedure

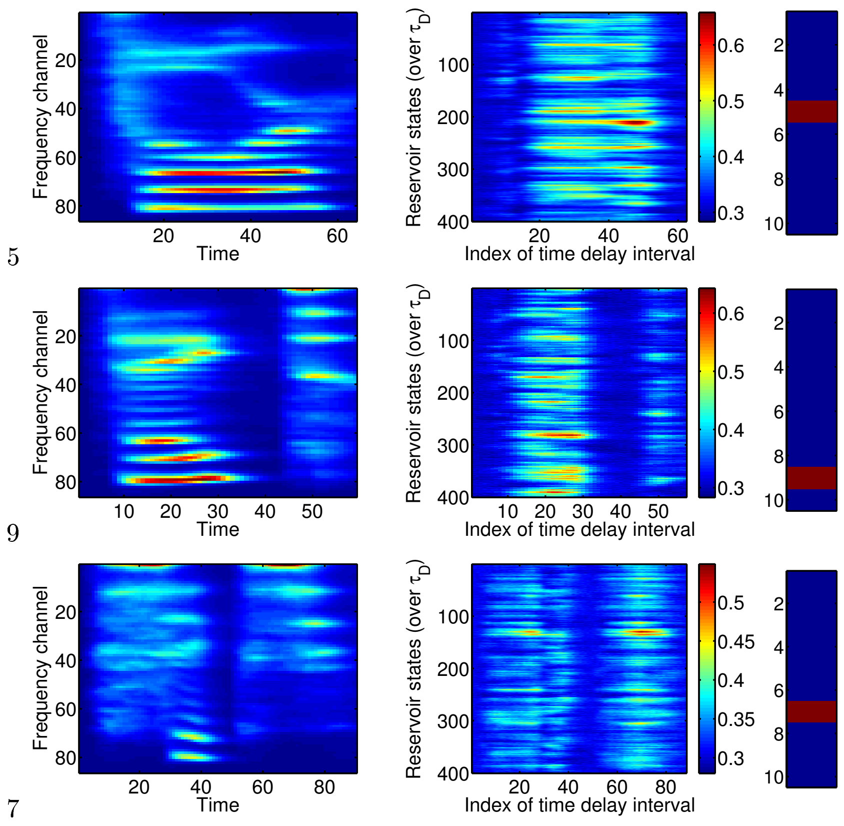

Fig. 9: Liquid State Machine data structure: inputs, reservoir states, expected read-outs (targets) (from left to right).

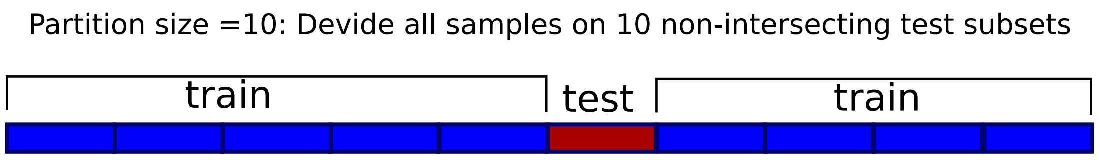
Partition size =10: Devide all samples on 10 non-intersecting test subsets
train
test
train

Fig. 10: Cross-validate partition.

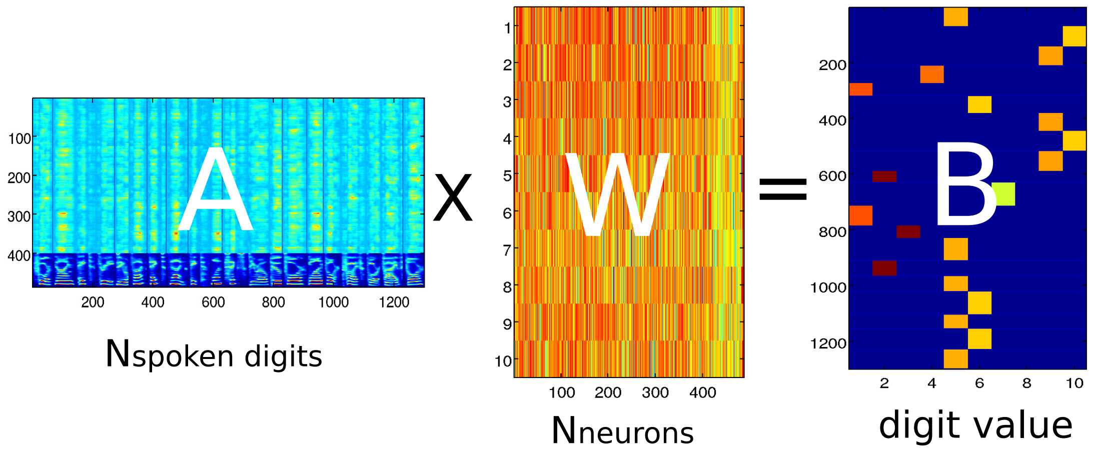

Fig. 11: Training

The following explanation why to use ridge regression instead of linear one is extracted from the PhD thesis of Verstraeten (Verstraeten, 2009).

Usually, linear models are regularized by inserting an additional penalty term - related to the norm of the weights - to the prediction on the training set. This can be expressed formally as:

$$W_{opt} = \arg\min_{W} \|AW - B\|_k + \lambda \|W\|_l \,,$$

where $\|\bullet\|_p$ denotes the p norm, and λ is a parameter that controls the tradeoff between the error on the training set and the weight norm. Depending on the values of k and l, different forms of regularized linear models can be obtained. For instance, if $k = 2$ and $l = 1$, we get what is called LASSO-regression (Tibshirani, 1996), which leads to sparse models (meaning that many of the weights are zero), which in turn could be useful in e.g. hardware implementations because of the limited number of connections. Recent results also show that for certain tasks LASSO-regression can achieve better results or more stable generators than classic linear regression (Dutoit et al., 2009).

The more common case, and the case we will discuss here and use in all the experiments described in this work, is the case where $k = l = 2$.

$$W_{opt} = \arg\min_{W} \|AW - B\|^2 + \lambda \|W\|^2 \,.$$

Here, both norms are the standard Euclidean norms and we obtain what is known as ridge regression or (less commonly) Tikhonov regression (Tikhonov and Arsenin, 1977). Here, as with standard least-squares regression, the solution to the equation can be obtained in one step as follows:

$$W_{opt} = (A^T A + \lambda I)^{-1} A^T B.$$

In this case, there exists a convenient relation between the regularization parameter and the model complexity (which is related to its computational expressive power). One can state:

$$\gamma(\lambda) = \sum_{i=1}^{p} \frac{\sigma_i}{\sigma_i + \lambda},$$

where λ is the value of the regularization parameter, $\gamma(\lambda)$ is the *number of effective parameters*, and σ_i is the i-th eigenvalue of the matrix $A^T A$. The effective number of parameters quantifies the number of free parameters of the model (the readout weights, in the case of reservoirs) are actually used, and as such gives an indication of the complexity of the model (Moody, 1992).

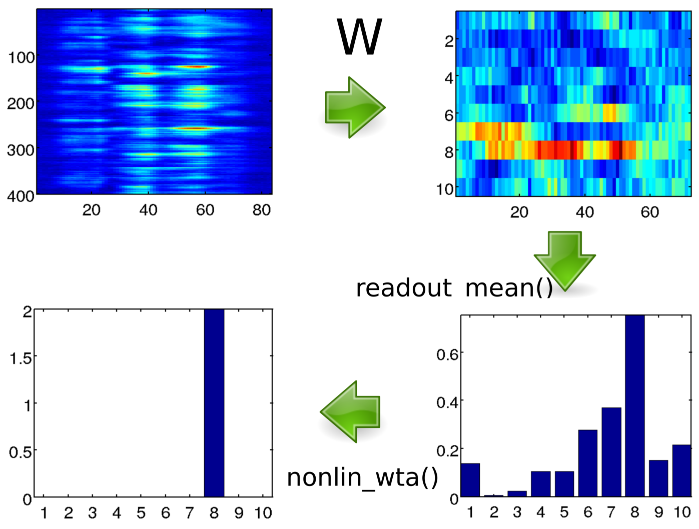

Fig. 12: Read-out

Another way of performing regularization is by adding noise to the data. The noise on the training set will reduce the absolute magnitude of the parameters, which in turn constrains the model complexity. In (wyffels et al., 2008a), an experimental evaluation of both regularization methods is done using a time series prediction and a system identification task. The conclusion is that both methods yield similar optimal results. However, ridge regression is preferable for pragmatic reasons: using noise as regularization mechanism is inherently non-deterministic (which means that results are not generally reproducible), and in cases where state noise is added to a reservoir with output feedback, the whole reservoir needs to be resimulated for every noise level, which dramatically increases the computational requirements.

While ridge regression offers good control over the smoothness of the obtained regression/readout function through a single parameter λ, the best value to use still needs to be determined. The optimal value for this regularization parameter would be the one that gives the best performance on unseen data. Determining this optimal value is done using grid-searching of the parameter with cross-validation.'

9 Read-out and evaluation

After training the matrix W_{opt} is calculated. Now one should apply this matrix to test samples and compare the read-outs with targets. The procedure of read-out is outlined in Fig.12. There are three main steps: 1) applying the W_{opt} and getting raw read-out; 2) applying first post-processing step, function `readout_mean()`, which in fact integrate (sum) the result over time axis (horizontal line); 3) applying function `nonlin_wta()` (*w*inner-*t*ake-*a*ll-non-linearity), which selects best (max) value from previously obtained elements. Final result of read-out is binary array with 0's everywhere except one element with1 which indicate the answer.

10 Scoring

The scoring is simple. After read-out every sample from test subset produce an answer. To collect all errors one should compare answers and targets. If they are equal, the error is 0. If not the error is 1. This procedure is depicted in Fig.13.

Total score collects all errors in test subset,

$$score = \frac{(0 + 1 + \cdots + 1 + 0)}{number\ of\ samples}$$

Finally Word Error Rate (WER) is just mean value of scores taken over all folds (cross-validate cycles),

$$WER = \text{mean}(score)$$

To find good and bad samples in speech digits database statistical analysis were done. The cross-validation process were performed for 500 different randomly chosen partitions. Numbers of samples which give error were recorded. Then statistical analysis were performed and resulted in histogram which

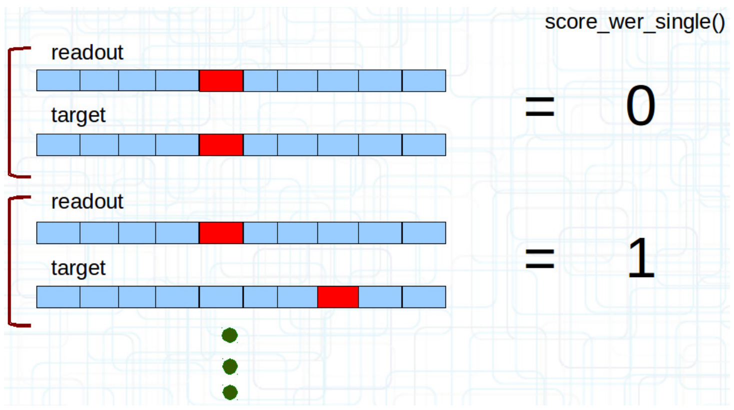

Fig. 13: Scoring

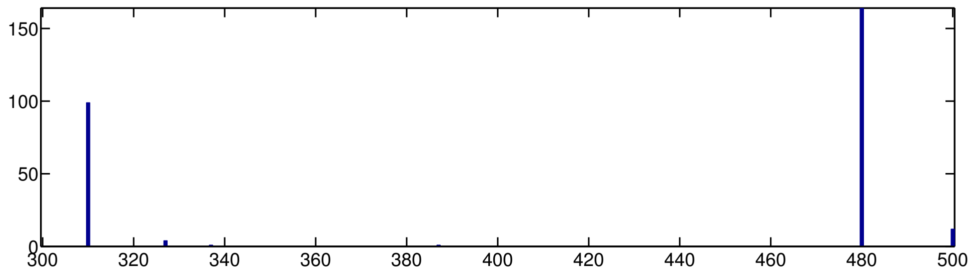

Fig. 14: Statistical analysis of samples which give error in score counting

is shown in Fig.14 The worst result demonstrate samples with numbers $n_{sample} = 480; 310; 500; 327$ with counts $count_{sample} = 164; 99; 12; 4$ from 500 runs. Two most error generating samples are with number 480 and 310. Their data (input and reservoir response) are depicted in Fig.15. Both of them represent digit 'Ten'. For comparison at the bottom line in the picture the sample with number 110 is shown. This sample never gives error in statistical analysis. Looking to the pictures it is difficult to understand why the simples 480 and 310 generate most errors in statistics. It is obviously that an additional investigation is needed. For example, one observation should be noted here. Every error-generating digit sample has its own miss-interpreted digit. In all cases when digit sample 480 produce an error it was miss-interpreted as '2'. The table of all miss-spelled associations is presented below. Moreover in the table closest competitor is indicated. In almost all cases it correspond to a correct digit.

Sample	Error digit	Competitor
480	2	10
310	8	10
500	2	10
327	8	7(10)

References

[1] Lyon, R. (1982). A computational model of filtering, detection and compression in the cochlea. In Proceedings of the IEEE ICASSP, pages 1282–1285.

[2] 'Auditory Toolbox' can be found here `http://cobweb.ecn.purdue.edu/~malcolm/interval/1998-010/`.

[3] MATLAB `Reservoir Toolbox` designed by Ghent University `http://reslab.elis.ugent.be/rctoolbox`

[4] Verstraeten, David (2009). Reservoir Computing: Computation with Dynamical Systems, Electronics and Information Systems, Gent, Ghent University, pp. 190 `http://organic.elis.ugent.be/sites/www.reservoir-computing.org/files/sites/organic.elis.ugent.be/files/dvrstrae_organic.pdf`

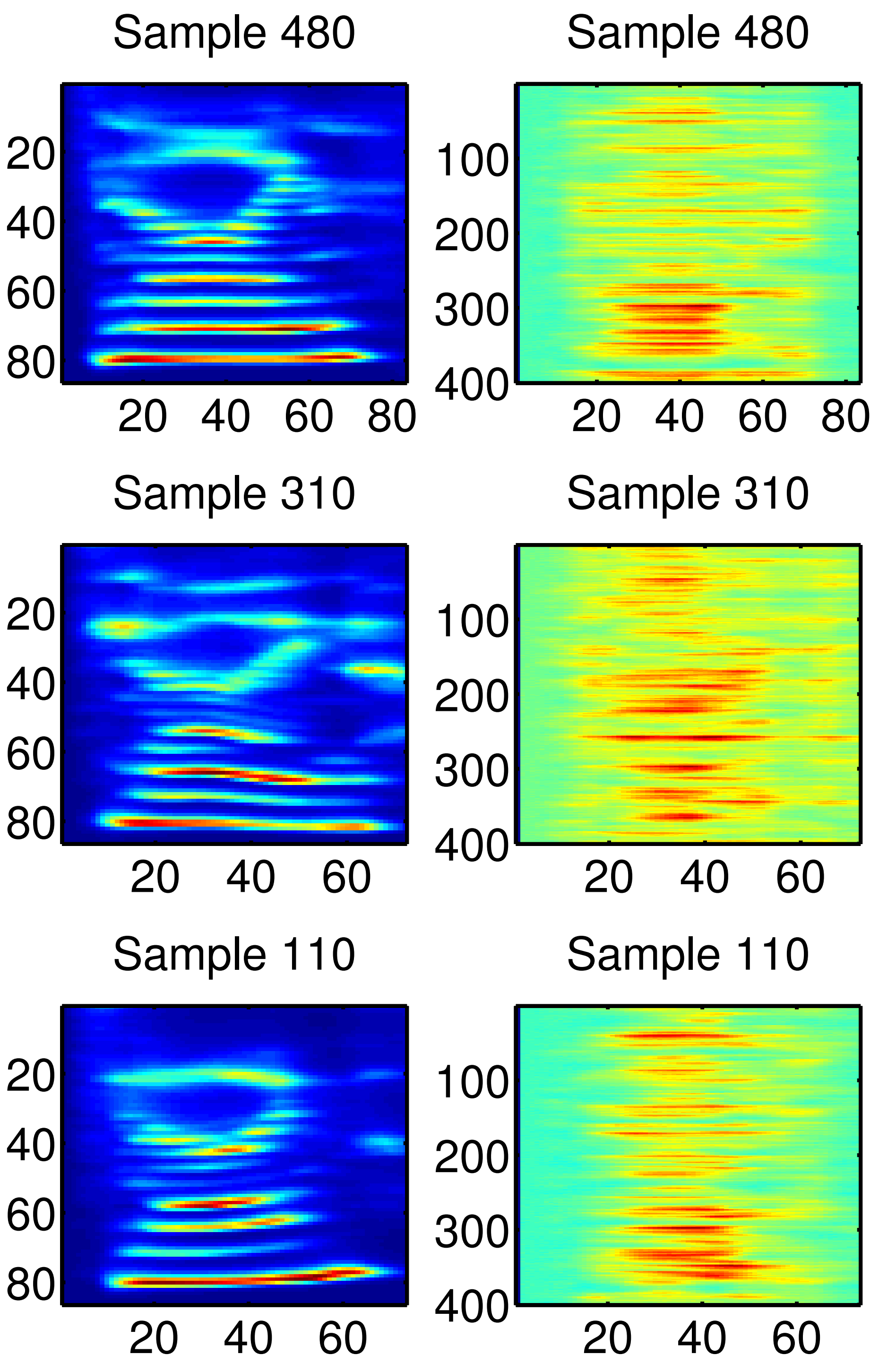

Fig. 15: Data structure for three samples with numbers 480, 310 which produce most errors in statistical analysis and 110 which never gives error.

[5] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. J. Royal. Statist. Soc B., 58(1):267–288.

[6] Dutoit, X., Schrauwen, B., Van Campenhout, J., Stroobandt, D., Van Brussel, H., and Nuttin, M. (2009). Pruning and regularization in reservoir computing. Neurocomputing, 72:1534–1546.

[7] Tikhonov, A. and Arsenin, V. Y. (1977). Solutions of Ill-Posed Problems. Winston and Sons.

[8] Moody, J. (1992). The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In Proceedings of NIPS, volume 4, pages 847–854.

[9] wyffels, F., Schrauwen, B., and Stroobandt, D. (2008a). Stable output feedback in reservoir computing using ridge regression. In International Conference on Artificial Neural Networks.