Feature Engineering and Stacked Echo State Networks for Musical Onset Detection

Peter Steiner*, Azarakhsh Jalalvand†, Simon Stone*, Peter Birkholz*
*Institute for Acoustics and Speech Communication
Technische Universität Dresden, Dresden, Germany
Email: peter.steiner@tu-dresden.de
†Ghent University – imec, *IDLab*, Ghent, Belgium
Email: azarakhsh.jalalvand@ugent.be

Abstract—**In music analysis, one of the most fundamental tasks is note onset detection – detecting the beginning of new note events. As the target function of onset detection is related to other tasks, such as beat tracking or tempo estimation, onset detection is the basis for such related tasks. Furthermore, it can help to improve Automatic Music Transcription (AMT). Typically, different approaches for onset detection follow a similar outline: An audio signal is transformed into an Onset Detection Function (ODF), which should have rather low values (i.e. close to zero) for most of the time but with pronounced peaks at onset times, which can then be extracted by applying peak picking algorithms on the ODF. In the recent years, several kinds of neural networks were used successfully to compute the ODF from feature vectors. Currently, Convolutional Neural Networks (CNNs) define the state of the art. In this paper, we build up on an alternative approach to obtain a ODF by Echo State Networks (ESNs), which have achieved comparable results to CNNs in several tasks, such as speech and image recognition. In contrast to the typical iterative training procedures of deep learning architectures, such as CNNs or networks consisting of Long-Short-Term Memory Cells (LSTMs), in ESNs only a very small part of the weights is easily trained in one shot using linear regression. By comparing the performance of several feature extraction methods, pre-processing steps and introducing a new way to stack ESNs, we expand our previous approach to achieve results that fall between a bidirectional LSTM network and a CNN with relative improvements of $1.8\,\%$ and $-1.4\,\%$, respectively. For the evaluation, we used exactly the same 8-fold cross validation setup as for the reference results.**

Index Terms—**Reservoir computing, echo state networks, note onset detection.**

I. Introduction

Note onset detection – detecting the beginning of new musically relevant note events in an acoustic signal – is often used as a starting point for related high-level-tasks, such as note beat tracking or tempo estimation. Furthermore, onset detection can support multipitch tracking [1], in which all active notes present in an audio signal should be detected. In [2], we have provided an extensive review on onset detection. The main outline for onset detection can be summarized as follows: An audio signal is transformed into an Onset Detection Function (ODF). In the ideal case, the ODF is (very close to) zero for most of the time. In case of onsets, the ODF has pronounced peaks. By applying peak picking algorithms on the ODF, the onset times are extracted. In recent studies [3]–[5], different neural networks such as bidirectional LSTM networks and CNNs have been introduced as state of the art in learning an ODF from input feature vectors. This led to F-Measures (defined later) of 0.873 and 0.903 on the Böck dataset [6]. The CNN approach [5] is the current state of the art.

In our previous study [2] that also includes a detailed review about onset detection, we have introduced Echo State Networks (ESNs) to learn the ODF from input feature vectors. The best result (F-Measure of 0.806) there was obtained with a bidirectional ESN with 8000 neurons. This was promising, but still far behind the state of the art.

In this paper, we build upon our preliminary results in [2] and expand it in several directions:

- We compare different approaches for standardization of the feature vectors.
- We explore the impact of using different window lengths and the concatenation of different feature vectors.
- We investigate whether the first and second derivatives as additional features both provide additional information for the ESN.
- We propose a novel way of stacking ESNs and show that this strongly improves the performance of onset detection.

The remainder of this paper is structured as follows. Section II presents a brief introduction the outline of onset detection with ESNs, introducing the proposed extensions. The experimental setup including the dataset used is described in Section III. Section IV continues with a detailed discussion of the different experiments conducted for this work, and the results are compared to the current state of the art. Finally, in Section V, we summarize our conclusions and give a brief overview about the future work.

II. Onset Detection with Echo State Networks

Echo State Networks (ESNs) [7] belong to the family of Recurrent Neural Networks (RNNs). In contrast to the typical iterative training procedures of RNNs and deep-learning architectures, in ESNs only a very small part of the weights is trained in one shot using linear regression.

The recurrent connections inside the reservoir, the core element of an ESN, can retain information from previous inputs for a certain amount of time. This so-called memory is tuned by the hyper-parameters of the reservoir and leads to a short or long-term memory. Typically, the neurons inside the

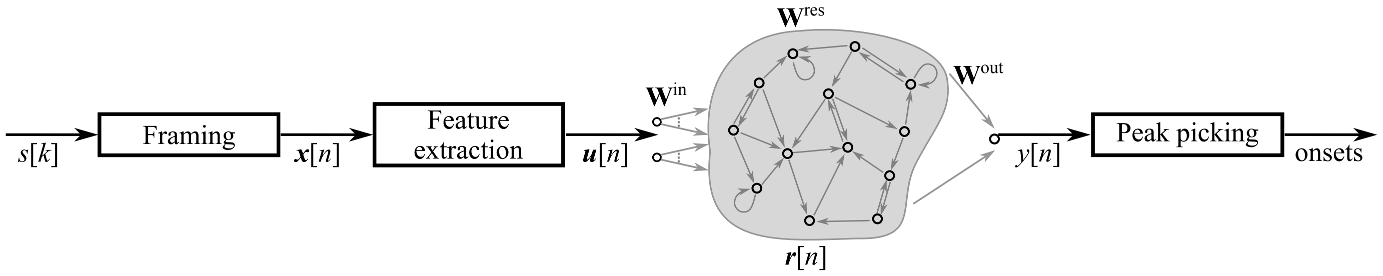


Fig. 1: Outline of the ESN-based proposed model from [2]: The mono input signal $s[k]$ with the sample index k was divided into overlapping frames, from which normalized feature vectors $\mathbf{u}[n]$ were extracted and fed into the reservoir using the input weight matrix $\mathbf{W}^{\text{in}}$. The reservoir consists of unordered and, via the reservoir matrix $\mathbf{W}^{\text{res}}$, sparsely connected neurons. The one-dimensional output $y[n]$ is a linear combination of the reservoir states $\mathbf{r}[n]$ and the output weight matrix $\mathbf{W}^{\text{out}}$, which was trained using linear regression. The output served as the ODF, in which onsets were extracted using a peak picking algorithm.

reservoir have non-linear activation functions, such as sigmoidal activations. Because the number of reservoir neurons is usually much higher than the number of input features, the reservoir transforms the low-dimensional input space non-linearly into a high-dimensional feature space, which facilitates the separation of the input space with hyper-planes. The desired output is thus a multi-linear function of the transformed features. The main outline of the proposed ESN-based model for onset detection is depicted in Fig. 1.

A. *Framing*

The mono input signal $s[k]$ with the sample index k and a sampling frequency of $44.1\,\text{kHz}$ was divided into overlapping frames with a frame rate of $100\,\text{Hz}$ and the frame index n. Each frame was windowed using a Hann window. In [3]–[5], feature were extracted using several window lengths of 1024, 2048 and 4096 without elaborating the motivation in details. In this paper, we explored the impact of different window sizes more in detail. Therefore, we extracted features in the following settings:

- We used window sizes of 1024, 2048 and 4096 samples individually to extract features. This allowed us to investigate, whether and how different window sizes can help detecting different types of onsets more robustly.
- Next, we concatenated the extracted feature vectors from different window sizes to obtain a much larger feature vector.

B. *Feature extraction*

As in [2], we computed the short-term Fourier transform of each windowed frame and applied a triangular filterbank with a logarithmic frequency spacing to every short-term spectrum to reduce the dimension of the feature vectors. In preliminary experiments we compared, how different numbers of filters per octave from 1 to 24 influence the performance of the ESN in terms of F-Measure. It turned out that a number of 7 filters per octave was the most suitable one. Finally, we applied the $\log_{10}$ to the magnitude plus 1 to avoid negative features. The range of the magnitude was thus bounded between 0 and around 2.

Onsets are strongly correlated with energy changes in frequency bands. In energy-based approaches for onset detection, such as [8], the "spectral flux" – the half-wave rectified first order difference along time – was used as a feature to obtain an ODF. We adapted this, and used a first order difference filter kernel with a size of 3 to compute the differences based on one feature vector before and after the current frame. Negative differences were set to zero, and this half-wave rectified vector appended to the features described above.

In phoneme recognition [9], the second derivative is often used as an additional feature, because it can capture dynamic information. In this work, we investigated whether the second derivative provides additional useful information to the ESN. The second derivative can be particularly useful to detect very soft onsets, as they are followed by a long attack phase. Thus we applied the same filter kernel again on the first derivatives computed before, and set negative values to zero. We also appended this half-wave rectified vector to the feature described before.

Before feeding the feature vectors in the ESN, we compared different standardization methods:

- No standardization at all
- z-score over time
- z-score over time and features
- Subtract about half the maximum value from all features and thus convert them from unipolar to bipolar values.

C. *Echo State Network (ESN)*

The main outline of an ESN is depicted in the center of Fig. 1. It basically consists of three weight matrices. The input weights $\mathbf{W}^{\text{in}}$ pass the input features to the reservoir, which is an unordered group of N^{res} neurons. The reservoir weights $\mathbf{W}^{\text{res}}$ connect the neurons inside the reservoir to each other. The output weights $\mathbf{W}^{\text{out}}$ connect the neurons inside the reservoir with the output nodes.

Both, $\mathbf{W}^{\text{in}}$ and $\mathbf{W}^{\text{res}}$, are initialized from random distributions. The reservoir weights $\mathbf{W}^{\text{res}}$ needs to fulfill the *Echo State Property* (ESP), which says that, for a finite input sequence, the reservoir states need to decay in a finite time [7]. This is done by normalizing $\mathbf{W}^{\text{res}}$ to its maximum absolute eigenvalue.

The key difference between ESNs and typical RNN architectures is that both, $\mathbf{W}^{\text{in}}$ and $\mathbf{W}^{\text{res}}$ are initialized, fixed and no more optimized during the training. Only the output weights $\mathbf{W}^{\mathrm{out}}$ are trained using linear regression.

With $\mathbf{r}[n]$ representing the reservoir state, Equations (1) and (2) are used to describe ESNs.

$$\begin{aligned}\mathbf{r}[n] =&(1-\lambda)\mathbf{r}[n-1]+ \\ &\lambda f_{\mathrm{res}}(\mathbf{W}^{\mathrm{in}}\mathbf{u}[n]+\mathbf{W}^{\mathrm{res}}\mathbf{r}[n-1]+\mathbf{w}^{\mathrm{bi}})\end{aligned} \tag{1}$$

$$y[n] = \mathbf{W}^{\mathrm{out}}\mathbf{r}[n] \tag{2}$$

Equation (1) is a leaky integration of the reservoir states $\mathbf{r}[n]$ with $\lambda \in (0,1]$ being the leakage, and $f_{\mathrm{res}}(\cdot)$ is the non-linear reservoir activation. We used the $\tanh$-function, because its lower and upper boundaries of ± 1 ensure stable reservoir states, and it is two times steeper than the sigmoid activation. Every neuron in the reservoir has a constant bias input from the bias weight vector $\mathbf{w}^{\mathrm{bi}}$, which is initialized and fixed from a uniform distribution between ± 1. Equation (2) shows that the one-dimensional output $y[n]$ is a linear combination of a given reservoir state $\mathbf{r}[n]$.

For training, all reservoir states are collected in the reservoir state collection matrix $\mathbf{R}$. To add the intercept term for linear regression, every reservoir state $\mathbf{r}[n]$ is expanded by a constant of 1. The desired outputs $d[n]$, which are 0 for non-onsets, 1 for onsets and 0.5 for frames around onsets, are collected into the desired output collection vector $\mathbf{D}$. Afterwards, $\mathbf{W}^{\mathrm{out}}$ is obtained using regularized linear regression (3), i.e. ridge regression to prevent overfitting to the training data. The regularization parameter $\epsilon = 0.01$ penalizes large values in $\mathbf{W}^{\mathrm{out}}$, and $\mathbf{I}$ is the identity matrix. The size of the output weight matrix $N^{\mathrm{out}} \times (N^{\mathrm{res}} + 1)$ determines the total number of free parameters to be trained in ESNs. The output $y[n]$ corresponds to the onset detection function ODF.

$$\mathbf{W}^{\mathrm{out}} = \left(\mathbf{R}\mathbf{R}^{\mathrm{T}} + \epsilon\mathbf{I}\right)^{-1}\left(\mathbf{D}\mathbf{R}^{\mathrm{T}}\right) \tag{3}$$

An ESN has several control parameters, which need to be tuned for onset detection: α_{u}, ρ, and α_{bi} control the absolute importance of the input feature vector, old reservoir state and the constant bias inputs, respectively. They are global scaling factors of the weight matrices $\mathbf{W}^{\mathrm{in}}$, $\mathbf{W}^{\mathrm{res}}$ and $\mathbf{w}^{\mathrm{bi}}$. The leakage λ is a control parameter for the leaky integration and adapts the input dynamics to the output dynamics.

The initialization of an ESN for onset detection has been studied more detailed in [2].

The basic ESN described so far, can be extended in several directions. Here, we focus on bidirectional reservoirs and propose a novel kind of stacked reservoirs.

1) Bidirectional reservoirs: In the case of bidirectional reservoirs, the input is first fed through the ESN and the reservoir states are collected as described before. Afterwards, the input is in reversed time and again fed through the ESN. These new reservoir states are again reversed in time and collected as described before. Finally, the reservoir states of both directions are concatenated and the output was computed using the concatenated reservoir states. This doubled the number of free parameters in W^{out}. For example, the number of parameters for a reservoir with 500 reservoir neurons in the bidirectional case is 500 in the forward and 500 in the backward path. The final training and computation of the output remained the same as before.

2) Stacked reservoirs: In the case of stacked reservoirs, the layers are trained sequentially using the same desired outputs in every layer. Typically, after fixing the hyper-parameters for one layer, its output can serve as the input for the next layer. By stacking reservoirs, the temporal modeling capacity of a single layer model is extended. This can be done for unidirectional as well as for bidirectional reservoirs.

In [9], [10], it was shown that this improved the results for phoneme or image recognition. However, those tasks are considerably different from onset detection, especially in the number of outputs. In those classification tasks, the second reservoir that received the output of the first reservoir, could determine a relationship between the different classes.

Thus, in this paper, we present a novel way of stacking reservoirs: In onset detection, we have a one-dimensional output, e.g. the ODF. Feeding the second reservoir just with one input provides only a narrow window of opportunities to improve the ODF. To overcome this problem, in this work, we fed the original feature vectors into the second reservoir, and used the ODF output of the first reservoir as a time-varying bias for the nodes of the second reservoir. Thus, each neuron in the second reservoir is still only connected to a limited number of K^{in} inputs and K^{rec} other neurons in the reservoir. Additionally, every node inside the second reservoir receives the ODF computed before, because the bias term is fully connected.

D. *Peak picking*

In [2], we have discussed that the output of an ESN after linear regression indicated an onset or non-onset, and would be zero for a non-onset and one for an onset, ideally. We used exactly the same peak picking algorithm as in our previous study. The algorithm itself was originally proposed in [5]. It is a simple threshold-based peak picking algorithm. At first, the ODF was smoothed using a Hamming window with 5 samples. Next, local maxima greater than a tunable threshold δ were detected. The locations of the resulting peaks were considered to be onsets.

III. Experimental setup

A. *Dataset*

For training and evaluation of the ESN models, we used the dataset introduced by Böck in [6]. It consists of around 102 min of audio files sampled at 44.1 kHz and 27 700 annotated onsets. The database is already split into eight folds for an 8-fold cross validation. We used six folds to train the ESN and one subset as a validation set to tune the hyper-parameters. Afterwards, we rotated the folds and repeated the optimization for another set of training and validation folds. This procedure was repeated eight times until every subset has been used for validation exactly one time. The hyper-parameters for the final evaluation were obtained by determining the lowest mean loss across all eight validation losses.

After fixing the hyper-parameters, the final model was trained using seven folds and tested on the last unseen fold. Again, this was repeated for eight times until each fold has been used for testing exactly one time. The results we report later are the mean values across the eight repetitions.

The dataset consists of all important types of onsets and various musical genres. Because the dataset was already used for several evaluations of algorithms for onset detection, we could directly compare the results of our cross-validation with state-of-the-art algorithms.

B. Measurements

As in [2], we compared our results with the state-of-the-art algorithms and report different measures using the `madmom`-library [11] with the same settings as used in [5]. The detected onset times were compared to the reference onset times. If an onset was detected in a time-window of $\pm 25\,\mathrm{ms}$ around a reference, it was considered as a true positive (TP). If no onset was detected in the window around a reference, it was considered as a false negative (FN). If any onset was detected outside the window, it was a false positive (FP). With these notations, the measurements precision P (Eq. (4)), recall R (Eq. (5)), F-Measure F (Eq. (6)) are defined. For a detailed description of the measurements, we refer to [2].

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{4}$$

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{5}$$

$$F = 2 \cdot \frac{P \cdot R}{P + R} \tag{6}$$

In this work, F served as the objective function to determine the peak picking threshold δ.

C. Implementation and optimization strategy

The algorithm was developed in Python 3 and will be available within a jupyter notebook in our Github repository[1] soon. Table I shows the hyper-parameters to be optimized. The optimization process was conducted using a sequence of grid and line searches. The optimization workflow to fix the ESNs hyper-parameters consisted of three steps:

1) We started with a grid search across α_u and ρ to determine a trade-off between forward and recurrent connections. Therefore, the other hyper-parameters were neutralized, i.e. no bias (α_bi = 0.0) and no leaky integration (λ = 1.0).
2) Next, α_bi was optimized without leaky integration (λ = 1.0).

[1]https://github.com/TUD-STKS/PyRCN

TABLE I: List over all hyper-parameters to be tuned. The values show the search range and the step size. We optimized the hyper-parameters in a sequence of grid searches.

Hyper-parameter	Range	Step
Input scaling α_u	$[0.1, 1.5]$	0.1
Spectral radius ρ	$[0, 1.0]$	0.1
Bias scaling α_bi	$[0, 10]$	0.1
Leakage λ	$(0.0, 1.0]$	0.1
Threshold δ	$[0.2, 0.6]$	0.02

3) Finally, λ was optimized.

For every parameter combination during this optimization workflow, the cosine distance between the target and the computed output was reported on the validation set. This was done separately for each of the eight folds.

After collecting the cosine distances from all eight folds, the mean cosine distance over all folds was computed. After each optimization step, the hyper-parameters leading to the lowest mean cosine distance were fixed and used for the optimization of the next parameter.

The reservoir size was fixed to 500 during this optimization process, and we used a unidirectional reservoir. It has been shown that the reservoir size tends to be independent from all other hyper-parameters [10], [12]. For the later evaluation, it was increased up to 28 000 neurons, and bidirectional reservoirs were used.

Considering the stacked ESN, the hyper-parameters were optimized layerwise. At first, the hyper-parameters for the first layer were fixed. Since the input features of the second layer were exactly the same as for the first, we just needed to tune α_bi and λ in the second layer.

After fixing all hyper-parameters, the peak picking threshold δ was determined by maximizing the F-measure using the training set of each fold.

IV. Results

In this sections, we present our results. We discuss the impact of feature engineering and different reservoir architectures on the onset detection performance.

A. Different standardization methods

To compare different standardization methods, we optimized single layer models with a window size of 2048 samples, because many algorithms for onset detection work with this size. Fig. 2 shows that the models trained without any standardization achieved reasonable results for different reservoir sizes. Computing the z-score either over time (which is the standard way to go in many publications) or over both time and feature achieved worse results. Without any standardization, the feature vectors were always non-negative and the maximum value was around 2. By subtracting 1 from the feature vectors, they were transformed into bipolar values between ± 1, which is exactly the linear range of the $\tanh$ non-linearities inside the reservoir. From Fig. 2 it can be seen that using this standardization worked almost equivalently well as no standardization at all. Of course, when doing further pre-processing, such as decorrelation, the z-score might be the best choice.

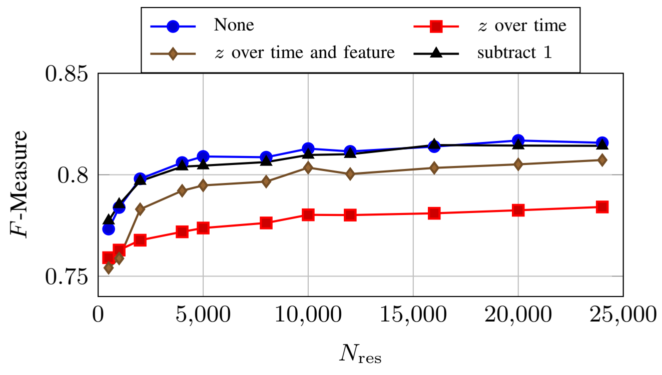


Fig. 2: Different ways of standardization: No standardization or subtracting 1.0 achieved nearly equal results. Any kind of statistical normalization removed important information from the features.

B. Different window sizes

To compare different window sizes, we optimized single layer models with the window sizes 1024, 2048 and 4096 samples. The current state of the art utilized a combination of those three window lengths. As can be seen in Fig. 3, the smallest window always leads to the lowest F-Measure. Furthermore, there is no real improvement when increasing the reservoir size N_{res}. A reason is that only relatively high f_0 values can be considered using the small window. However, in music, the f_0 can reach quite low frequency bands. Thus, larger windows led to significant improvements for every reservoir size.

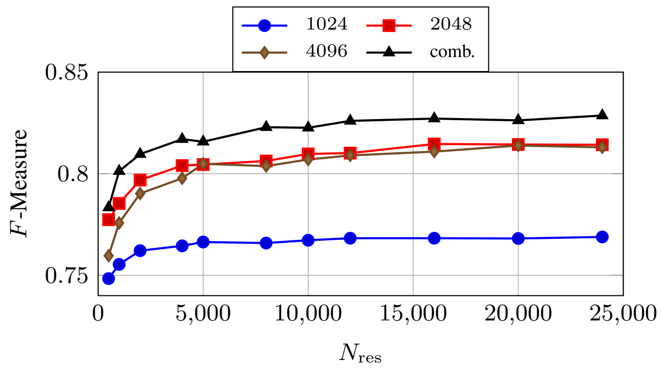


Fig. 3: Detection results using different window sizes: A size of 1024 samples alone is not suitable for general onset detection. At least for larger reservoirs, window sizes of 2048 and 4096 performed equally. Combining several window sizes performed much better.

Especially when increasing N_{res}, there is almost no difference between the window lengths 2048 and 4096. Our investigation on tuning the model based on the kind of onset (hard and soft) showed that shorter windows are suitable for processing percussive instruments, while a model trained with a window length of 2048 was able to deal with a wide range of tonal instruments. Very soft onsets, as they occur e.g. for singing voice or bowed string instruments, were recognized best by the model trained with a window length of 4096 samples.

If we concatenate the features extracted with the different window length, we obtain a larger feature vector that contains enough information to process different kinds of onset types. Thus, we can see in Fig. 3 that the models using this extended feature vector achieved the best results. The state of the art also used the same combination of different window lengths.

It is important to notice that, in contrast to typical RNN models, the number of free parameters to train ESNs does not increase with larger numbers of features. Instead, the number of trainable parameters just depends on the reservoir size and the number of output nodes. This makes it rather efficient to develop advanced features with ESNs, because a larger feature vector does not require a larger amount of training data.

C. Additional second derivative

The second derivative as an additional feature is useful in case of rather small reservoirs. As can be seen in Fig. 4, the F-Measures with and without the second derivative get similar if the reservoir size is increased and bidirectional. Our conclusion is that a useful variant of the second derivative can be learned by the reservoir itself and does not provide much additional information.

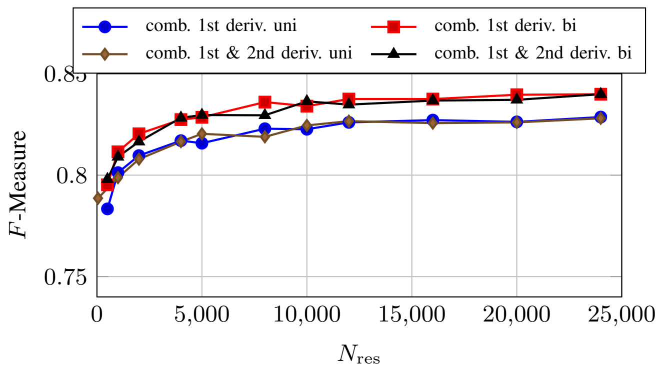
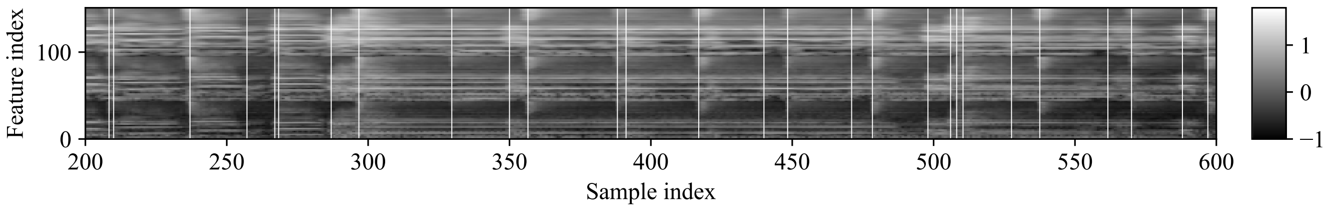


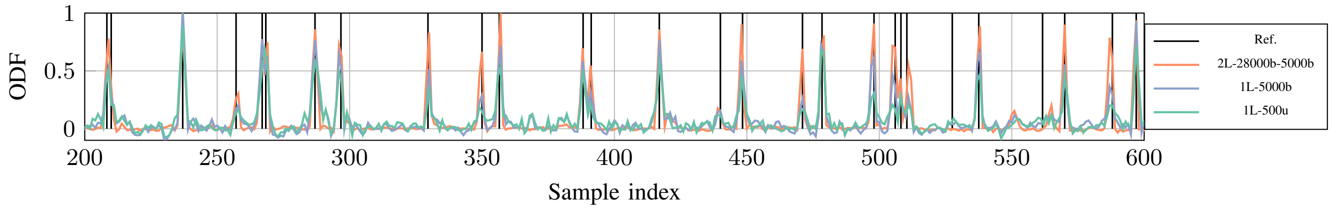
Fig. 4: For small reservoirs, the second derivative as additional feature gives the reservoir more information to be processed. However, in case of larger reservoirs, a more useful variant of the second derivative can be learned by the reservoir. Using a bidirectional instead of a unidirectional reservoir always increased the F-Measure clearly.

D. Unidirectional vs. bidirectional reservoirs

The state of the art utilized bidirectional LSTM networks or CNNs, which both incorporate future information. We compared the impact of supplying additional future information to a unidirectional model for combined window lengths of



(a) Spectrogram



(b) ODFs for different reservoir architectures

Fig. 5: Spectrogram and the ODFs for different N^{res} and stacked reservoirs. It can be observed that increasing the reservoir size and stacking reservoirs both improves the ODF.

1024, 2048 and 4096 samples. As can be seen in Fig. 4, this always increased the F-Measure clearly. We analyzed, which onset type benefited most from incorporating future information. It turned out that, once again, soft onsets were detected better with bidirectional architectures. Because the attack phase of soft onsets is much longer, we need to provide the ESN with additional temporal information. While this failed with the additional second derivative, passing the input sequence backwards through the ESN, we can provide the ESN model with a lot of additional information.

E. Stacked reservoirs

So far, the features were optimized, and we have shown that bidirectional reservoirs, who capture future information, are performing better than unidirectional reservoirs. From Fig. 4 we noticed that a reservoir size of 24 000 neurons did not yet lead to overfitting. Thus, we further increased the reservoir size to 28 000 neurons and added a second reservoir with 5000 neurons, which receives the original feature vectors and the ODF computed by the first reservoir as bias. We have compared the performance of different reservoir sizes in the second layer, and 5000 turned out to be the best performing reservoir size.

From Table II, we can see a strong impact of the second reservoir: While the best F-Measure with a single reservoir has quite an imbalanced ratio of P and R, the second reservoir is especially able to increase the Recall. Thus, many missed onsets, which are likely caused by too small peaks in the ODF, are recognized now. Those small peaks are strongly emphasized by the second reservoir, in which every neuron receives the previous ODF as a bias.

F. State of the art

Comparing our approach with the state of the art, we can see in Table II that the model consisting of a stacked reservoir outperformed the bidirectional LSTM network proposed in [3] by ≈ 0.013, but fell short by ≈ 0.017 to the CNN proposed in [5]. Comparing the number of trainable parameters, much more data is required to train the CNN. Because of the layer-wise optimization for ESNs, just the weights for one layer need to be trained jointly. Furthermore, the number of trainable parameter does not depend on the size of the feature vectors, which allowed it to simply concatenate features extracted from different window sizes. Thus, the ESN models proposed here have more free parameters than the bidirectional LSTM network, but much less than the CNN.

TABLE II: P, R and F for large reservoirs evaluated using the 8-fold cross validation. The reference models were evaluated on the same dataset by the authors. Our best model (2L-28000b-5000b) has outperformed the bidirectional LSTM network [3] and fell short to the CNN [5].

Architecture	P	R	F	Parameters
1L-24000b	0.881	0.804	0.840	48 001
2L-28000b-5000b	**0.920**	0.855	0.886	66 002
ESN [2]	0.854	0.774	0.812	16 001
Bidirectional LSTM [3]	0.892	0.855	0.873	20 225
CNN [5]	0.917	**0.889**	**0.903**	289 406

The training time for the best performing model was around 600 min, measured on a single 2.5 GHz core with around 10 GB RAM.

G. Reservoir analysis

We have seen that the F-Measure strongly depends on the reservoir size. To analyze the behavior of different reservoirs to some extent, an excerpt of the example song “ff123_ATrain” from the test set is visualized in Fig. 5. The upper plot (Fig. 5a) shows the concatenated feature vectors obtained from the different window lengths 1024, 2048 and 4096 samples. The bottom third was extracted with a window length of 4096, the middle third with a window length of 2048, and the top third with a window length of 1024 samples. The vertical white lines mark the reference onset positions. We can clearly observe that different onsets can be captured better with different window lengths. This supports the advantage of concatenating the feature vectors from different window lengths.

Fig. 5b shows the corresponding ODFs, computed using a unidirectional reservoir with 500 neurons (1L-500u), a bidirectional reservoir with 5000 neurons (1L-5000b), and with a layered architecture of two bidirectional reservoirs with 28 000 neurons in the first and 5000 neurons in the second layer (2L-28000b-5000b).

Comparing the ODFs computed by the 1L-500u and the 1L-5000b architectures, we observed that many weak peaks, such as around the sample 500, get more prominent in case of the larger bidirectional architecture (1L-5000b). However, the peak around the sample 530 was not improved, but became less prominent instead.

A very strong improvement can be noticed by stacking reservoirs. As we noticed in the orange line plot of Fig. 5b that refers to the architecture (2L-28000b-5000b), the ODF is very clear now, for example around the samples 270, 500 and 570. Only a few spurious peaks occurred at non-onset positions. One negative example is the peak around the sample 550, which is higher than before. As this peak has already appeared in the ODFs of the one-layer architectures, the two-layer architectures has emphasized it strongly.

Overall, a second reservoir that receives the feature vectors together with a useful ODF clearly improves the results from a single layer architectures.

V. Conclusions and Outlook

We showed in detail, how we can develop useful features for onset detection using ESNs, as we have compared several ways of feature normalization and the impact of different window sizes. The z-score, which is often used and recommended as a pre-processing step, is useful if all features do carry useful information for the specific task. However, when working with spectrograms or filterbank outputs, we cannot ensure that all features contain task-specific information. In that case, the z-score might over-emphasize exactly those features that do not carry task-specific information.

We have shown how different window sizes are useful for different kinds of onsets. Short windows are useful in case of hard onsets with a quick attack phase, whereas long windows are more useful to analyze soft onsets with a rather long attack phase. We showed and elaborated the significant impact of extracting and concatenating features using three different window sizes of 1024, 2048 and 4096. One take-away from the extended feature vectors is that the number of trainable parameters for ESNs is independent of the feature size. This is different to typical RNN architectures, in which the number of free parameter strongly depends on the size of the feature vector.

We have investigated the second derivative as an additional feature that could potentially be able to detect soft onsets with a relatively long attack phase. This only improved the results for very small reservoirs. We conclude that large reservoirs already learn a useful variant of the second derivative by themselves.

Furthermore, we have introduced a new kind of stacked reservoirs by using the ODF computed in one reservoir as the bias term for the next one, which receives the original feature vectors as input. This makes it possible for the second reservoir to improve the ODF a lot, and offers a lot of room for further investigations: In speech and digit recognition with ESNs [10], [13], more than two reservoirs were stacked, which always improved the results and made it possible to use less neurons per layer. We will investigate, whether the onset detection also benefits from additional layers. The original ESN by Herbert Jaeger includes an additional feedback term with connections from the output node back to the reservoir neurons. This is related to the bias term in the second reservoir. We will incorporate feedback and investigate the impact of feedback connections. If the feedback it works as the bias in the second reservoir, we expect improvements in onset detection together with a reduced number of trainable parameters and significantly less training and test time. Currently, the training time for the best performing model on a single 2.5 GHz core with around 10 GB RAM takes around 600 min. If this amount of training time and the number of free parameters can be reduced, it would be an important step to include ESNs in real-world applications.

It is, basically, also possible to combine different classifiers. We can, for example, use the output of a CNN as a bias for an ESN in the second layer.

As a future work, one can also combine the systems for onset detection and multipitch tracking [14], which is an important step towards a system for Automatic Music Transcription. Given that related high-level-tasks, such as beat tracking or tempo estimation are working with similar peaky target functions, it would be interesting to investigate, whether the presented approach can be adapted to such tasks.

Acknowledgements

The parameter optimizations were performed on a Bull Cluster at the Center for Information Services and High Performance Computing (ZIH) at TU Dresden.

This research was financed by Europäischer Sozialfonds (ESF) and the Free State of Saxony (Application number: 100327771) and Ghent University.

Thanks to Sebastian Böck to providing the library `madmom` [11].

References

[1] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, “Onsets and frames: Dual-objective piano transcription,” 2017.

[2] P. Steiner, S. Stone, and P. Birkholz, “Note onset detection using echo state networks,” in *Studientexte zur Sprachkommunikation: Elektronische Sprachsignalverarbeitung 2020*, R. Böck, I. Siegert, and A. Wendemuth, Eds. TUDpress, Dresden, 2020, pp. 157–164.

[3] F. Eyben, S. Böck, B. W. Schuller, and A. Graves, “Universal onset detection with bidirectional long short-term memory neural networks,” in *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010, Utrecht, Netherlands, August 9-13, 2010*, 2010, pp. 589–594. [Online]. Available: http://ismir2010.ismir.net/proceedings/ismir2010-101.pdf

[4] J. Schlüter and S. Böck, “Musical onset detection with convolutional neural networks,” in *6th International Workshop on Machine Learning and Music (MML), Prague, Czech Republic*, 2013.

[5] ——, “Improved musical onset detection with convolutional neural networks,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 6979 – 6983.

[6] S. Böck, F. Krebs, and M. Schedl, “Evaluating the online capabilities of onset detection methods,” in *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012*, 2012, pp. 49–54. [Online]. Available: http://ismir2012.ismir.net/event/papers/049-ismir-2012.pdf

[7] H. Jaeger, “The ”echo state” approach to analysing and training recurrent neural networks,” German National Research Center for Information Technology, Tech. Rep. GMD Report 148, 2001. [Online]. Available: http://www.faculty.iu-bremen.de/hjaeger/pubs/EchoStatesTechRep.pdf

[8] S. Dixon, “Onset detection revisited,” in *Proceedings of the International Conference on Digital Audio Effects (DAFx-06)*, Montreal, Quebec, Canada, Sept. 18–20, 2006, pp. 133–137. [Online]. Available: http://www.dafx.ca/proceedings/papers/p_133.pdf

[9] F. Triefenbach, A. Jalalvand, B. Schrauwen, and J. pierre Martens, “Phoneme recognition with large hierarchical reservoirs,” in *Advances in Neural Information Processing Systems 23*. Curran Associates, Inc., 2010, pp. 2307–2315. [Online]. Available: http://papers.nips.cc/paper/4056-phoneme-recognition-with-large-hierarchical-reservoirs.pdf

[10] A. Jalalvand, K. Demuynck, W. D. Neve, and J.-P. Martens, “On the application of reservoir computing networks for noisy image recognition,” *Neurocomputing*, vol. 277, pp. 237 – 248, 2018, hierarchical Extreme Learning Machines. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231217314145

[11] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, “Madmom: A new python audio and music signal processing library,” in *Proceedings of the 24th ACM international conference on Multimedia*. ACM, 2016, pp. 1174–1178.

[12] F. Triefenbach, K. Demuynck, and J.-P. Martens, “Large vocabulary continuous speech recognition with reservoir-based acoustic models,” *IEEE Signal Processing Letters*, vol. 21, no. 3, pp. 311 – 315, March 2014.

[13] F. Triefenbach, A. Jalalvand, K. Demuynck, and J.-P. Martens, “Context-dependent modeling and speaker normalization applied to reservoir-based phone recognition,” in *Proc. Interspeech 2013*, 2013, pp. 3342–3346.

[14] P. Steiner, S. Stone, P. Birkholz, and A. Jalalvand, “Multipitch tracking in music signals using echo state networks,” in *2020 28th European Signal Processing Conference (EUSIPCO)*, 2020, pp. 126–130. [Online]. Available: https://www.eurasip.org/Proceedings/Eusipco/Eusipco2020/pdfs/0000126.pdf