Canary Song Decoder: Transduction and Implicit Segmentation with ESNs and LTSMs

Nathan Trouvain, Xavier Hinaut

Canary Song Decoder: Transduction and Implicit Segmentation with ESNs and LTSMs

Nathan Trouvain[1,2,3][0000−0003−2121−7826] and Xavier Hinaut[1,2,3,*][0000−0002−1924−1184]

[1] INRIA Bordeaux Sud-Ouest, France.
[2] LaBRI, Bordeaux INP, CNRS, UMR 5800.
[3] Institut des Maladies Neurodégénératives, Université de Bordeaux, CNRS, UMR 5293.
*Corresponding author: xavier.hinaut@inria.fr

Abstract. Domestic canaries produce complex vocal patterns embedded in various levels of abstraction. Studying such temporal organization is of particular relevance to understand how animal brains represent and process vocal inputs such as language. However, this requires a large amount of annotated data. We propose a fast and easy-to-train transducer model based on RNN architectures to automate parts of the annotation process. This is similar to a speech recognition task. We demonstrate that RNN architectures can be efficiently applied on spectral features (MFCC) to annotate songs at time frame level and at phrase level. We achieved around 95% accuracy at frame level on particularly complex canary songs, and ESNs achieved around 5% of word error rate (WER) at phrase level. Moreover, we are able to build this model using only around 13 to 20 minutes of annotated songs. Training time takes only 35 seconds using 2 hours and 40 minutes of data for the ESN, allowing to quickly run experiments without the need of powerful hardware.

Keywords: Birdsong · Echo State Networks · Long Short Terms Memory · RNN · Audio Classification · MFCC

1 Introduction

Birdsongs are a common resource to study sensorimotor learning of complex sequences of gestures. Many songbirds species, like the Bengalese finch or the canary, organize their songs on top of small stereotypical units, called *notes* or *syllables*. In the case of canaries, songs are composed of around 10 to 30 different syllables classes [11]. Individuals in a canary population might share syllables to some extent, due to the learning process of these vocalizations involving imitation from tutors, but each individual's repertoire is a unique combination of syllable types [18]. Canaries chain syllables of a same class together in repetitive patterns to form *phrases* (Figure 1), that are then chained to form songs. Phrase length (i.e. the number of repetitions of a syllable within a phrase) is uncorrelated to syllable type, but might play a role in song syntax [12]. Markowitz *et al.* [12]

also demonstrated the existence of a complex temporal structure in canary songs, where the order of phrases can be approximated by a high order Markov chain. This makes canary songs of a particular relevance to understand the mechanisms underlying the sequential organization of gestures in animal brains.

However, investigating the origin of the sequential organization of bird songs requires large amount of data. Studies like [12] should be more numerous in order to decipher general syntactic rules in canary songs, but this is limited by the available datasets of annotated song recordings. The annotation process, which usually consists in hand-labelling each phrase or syllable, is long and error prone. Automatic methods for song annotation of various bird species have been developed in the past years, to try leveraging the large audio recordings available. These methods use a large variety of techniques, like Dynamic Time Warping (DTW) [16, 9, 1], Hidden Markov Models (HMM) [3], Support Vector Machines (SVM) [15], unsupervised clustering [14] or combinations of Convolutional Neural Networks (CNN) and HMM [10]. While they can usually be applied on birdsongs where vocal patterns can be easily segmented like Bengalese finches, these methods are known to fail on birdsongs with more complex temporal patterns like canary songs, where segmentation is usually done by hand with only partial automation possible. Ongoing work described in Cohen *et. al* [4] solves the segmentation problem by using machine learning techniques to try to extract the position of syllables on spectrograms of the song while classifying them using CNN and Long Short Term Memory Recurrent Neural Networks (LSTM RNN).

We propose a method based on RNN processing spectral features to segment and classify canary songs. However, unlike [4], we focused on segmenting canary songs at phrase level, which is supposed to be sufficient for analysis like [12]. We compare simple neural architectures like a single LSTM or Echo States Networks (ESN) [7]. We chose these methods to demonstrate that lightweight algorithms, in comparison to more intensive deep learning methods, can be successfully used on difficult tasks similar to speech recognition, while being less time and energy consuming. Our canary dataset was composed of around 30 different labels (i.e. syllables types). Importantly, the limited number of parameters that need to be learned (compared to more complex architectures like [4]) enables one to apply them on limited amount of data while not overfitting on it; this is particularly interesting because it limits the amount of data necessary to be hand-labelled.

2 Methods

2.1 Song transduction task

Our task lies at the level of phrase annotation: each phrase is attributed a label corresponding to the syllable type that composed it, and each phrase label is delimited in time (expressed in seconds relative to the beginning of the audio sequence). An annotation is the combination of these absolute timestamps and of a syllable label. Figure 1 gives an example of an annotated portion of song, with groups of syllables (gray and white boxes) identified with arbitrary letters.

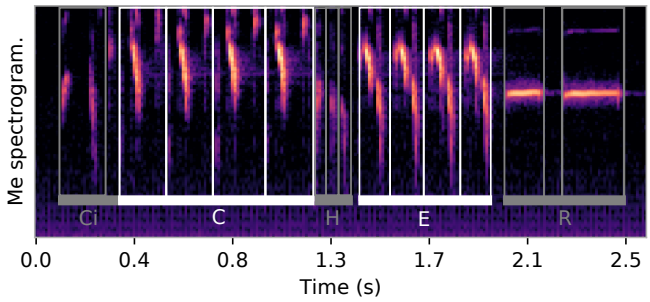

Fig. 1. Labeled canary song excerpt. Canary songs are made of a sequence of phrases, which are themselves made of repetitions of syllables. Each box delimits a *syllable*. The sequence of phrases is here *Ci-C-H-E-R* (sequence of labels on the figure). Sound is represented as Mel-spectrogram (i.e. spectrogram with Mel scale).

The task of canary song transduction can then be described as two *sequence-to-sequence* subtasks :

- an **audio-to-frame** transduction task – each time frame (each time point in figure 1) representing the preprocessed audio signal must be annotated with a corresponding phrase label.
- an **audio-to-phrases** transduction and segmentation task – the song must be transcribed as a sequence of tokens representing the phrase labels in the same fashion as the available dataset. In figure 1, the sequence of phrases to reconstruct is *Ci-C-H-E-R*.

To evaluate the fulfillment of these subtasks, we use two different metrics. The audio-to-frame task is evaluated using the frame accuracy (ACC) [2]. The ACC score is computed by assessing the correctness of the predicted labels for each time frame of data representing the song. Additionally, we also compute a global macro averaged *F1*-measure over all the annotated songs to take into account the dataset imbalance (i.e. the *F1*-measure is computed for all labels and then their unweighted mean is computed).

The audio-to-phrases task is evaluated using the word error rate (WER) measure at the phrase level (i.e. a "word" in this context is considered to be a phrase). The WER normalises the number of editions, deletions and substitutions necessary to align perfectly a predicted sequence with the correspondent expected one. It is thus defined as the Levenshtein distance between the predicted and expected sequences divided by the exact number of phrases in the expected sequence. All scores in section 3 are presented as mean $\pm$ standard deviation, figures included (plain areas of color represent standard deviation boundaries, curves represent mean).

We propose to train two RNN architectures on this task: an ESN (section 2.4) and an LSTM (section 2.5).

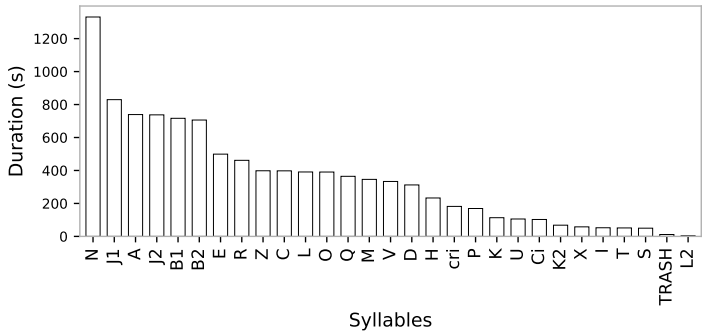

Fig. 2. Cumulated duration of the different types of phrases in the dataset.

2.2 Available data

We use a corpus of 459 songs recorded from a single canary, for a total song duration of approximately 3 hours 20 minutes. The songs were annotated by one human experimenter, and checked by other experimenters when the syllable category was unclear. They were then corrected by a second human experimenter assisted by early versions of the models presented in 2.4. The corrected version of the dataset contains 27 different types of phrases, each identified by a unique arbitrary label. We finally added three other classes of phrases. A *cri* class (*call* in French) is used to annotate all canary vocalizations that are not part of a sound, and are simple calls. A *SIL* (silence) class is used to annotate all segments where the bird is not singing. A *TRASH* class is used to annotate phrases that are impossible to classify clearly.

The phrases distribution is highly unbalanced inside the dataset, as presented in figure 2. As we want the model trained to be as good as possible at recognizing all types of syllables, we split the dataset in order to ensure that all types of syllables appear at least once in the training dataset. Once this condition is filled and the training set contains representative songs, we keep splitting the dataset until the training set contains 368 songs (around 2 hours 39 minutes), and the testing set contains 91 songs (around 41 minutes). We will make the dataset available on Zenodo (zenodo.org).

2.3 Data preprocessing

Speech recognition tasks generally need to convert raw audio signals to a format having a good trade-of between accuracy and number of features. We chose Mel-Frequency Cepstral Coefficients (MFCC) as a representation for canary songs: they allow to extract frequency features of the audio signal in a biologically meaningful way, while compressing the spectral representation down to a few coefficients. We mainly based our preprocessing steps on human audio standards (making some adjustments based on the frequency bands of canary vo-

calizations), assuming that we only need to extract the information that could have been perceived by human annotators. Indeed, canary syllables often form a clear identifiable pattern on a spectrogram, because it does not include complex harmonics.

MFCC computations where performed using the *Librosa* [13] Python library. Song spectrograms are first extracted using Short Time Fourier Transform every $11ms$ (often called *frame stride*) and computed on overlapping windows of $23ms$ (often called *window width*)[4], using a Hanning window to reduce edge effects. Then, we set the frequency range of a 128 filters Mel filterbank to $[500Hz; 8kHz]$, as canaries vocal patterns occur below $8kHz$ and as the $[0Hz; 500Hz]$ bandwidth represents mostly noise. Mel filters are appliyed on spectrograms through a dot product, and a final Discrete Cosine Transform (DCT) creates cepstral representations. We extracted 13 cepstral coefficients as usually fed to MFCC-based speech recognition models. We also computed the first (Δ) and second (Δ^2) derivatives of the MFCC signal, in order to provide the models with gestures dynamics. We therefore feed all models with a total of 39 features per timeframe. No normalization is applied to the MFCC representations, except a cosine liftering as described in [8], with a factor of 40, which helps to linearize the variance of the coefficients. We did not apply any other normalization to avoid any loss in representativeness in the extracted features.

2.4 Presentation of the Echo State Network model

We used ESNs with leaky integrator sigmoid neurons as transducers of the songs audio signals. ESNs can be described as randomized RNNs using simple learning rules to update the parameters of a *readout* layer of neurons. Appart from this readout layer of neurons, no other parameters are learned. All the other parameters are randomly chosen to build the *reservoir* and the input layer, and are kept fixed during the whole life cycle of the network. The reservoir is the main component of ESNs. It is a randomly, sparsely connected pool of neuronal units in charge of unfolding the temporal dynamics of the input data in a high dimensional space. The reservoir is randomly, sparsely connected to the sequential input stream of data through connections defined in the input layer. Activities (also known as *states*) of neuronal units in the reservoir are described following the equation 1 :

$$x[t] = (1-\alpha)x[t-1] + \alpha \tanh(\mathbf{W^{in}} \cdot u[t] + \mathbf{W} \cdot x[t-1]) \quad (1)$$

where $x[t]$ is the vector storing the activities of the N neurons at time t, and $u[t]$ is the current input vector at time t. The parameter α is the leaking rate (LR), controlling the time constant of the ESN, and set to 9×10^{-2}. $\mathbf{W} \in \mathbb{R}^{N \times N}$ stores connections weights of reservoir neurons, with $N = 1000$. All weights are randomly initialized from a uniform distribution using the method described

[4] *frame stride* and *window width* are respectively called *hop_length* and *win_length* in *librosa*.

by [5] with a spectral radius of 0.7 and a connection density of 20% (i.e. non-zero weights). $\mathbf{W^{in}} \in \mathbb{R}^{I \times N}$ stores connections weights going from the I-dimensional inputs to the N-dimensional reservoir. These weights are randomly chosen in $\{-1, 1\}$, with a connection density of 20%, and are then scaled by a constant factor called input scaling (IS). Because the input data is composed of three different sets of features (MFCCs, Δ, Δ^2) with different distributions, we chose to apply three different ISs to the corresponding connections weights. We therefore define the MFCC input scaling (ISS), the Δ input scaling (ISD), and the Δ^2 input scaling (ISD2), respectively set to 10^{-3}, 5×10^{-3} and 5×10^{-3}. Finally, we use a linear regression with $L2$ regularization (also called Tikhonov regression or *ridge*), with a regularization coefficient of 10^{-4}, to use the same as the LSTM model described in 2.5, to learn the $\mathbf{W^{out}} \in \mathbb{R}^{N \times U}$ matrix. The latter stores the readout connections between the N-dimensional reservoir and the U-dimensional output space. In our case, U is set to 30, the number of phrase labels. Outputs of the model are then computed as described in equation 2:

$$\hat{y}[t] = \mathbf{W^{out}} \cdot x[t] \tag{2}$$

where $\hat{y}[t]$ is the prediction vector at time step t, given the reservoir state $x[t]$. The predicted class index $c[t]$ in the repertoire is then given by 3:

$$c[t] = \text{argmax } \hat{y}[t] \tag{3}$$

All hyperparameters presented were optimized during a random search made on a subset of 100 songs including all syllable types.

2.5 Presentation of the LSTM model

We used a single LSTM (with forget gate) as originally presented in [6]. We set the number of units inside the LSTM to 72, in order to make the comparison fair with ESNs in term of total number of trainable parameters, which is around $30,000$. After adding a fully connected layer of units with softmax activation function on top of the LSTM to outputs class belonging probabilities, the model has a total of $34,446$ parameters. A $L2$ regularization is applied to the weights of the fully connected layer, with a regularization coefficient of 10^{-4}. We then trained the LSTM using Adam gradient descent algorithm, with a learning rate of 10^{-3}. Loss was computed using a log cross-entropy measure. Training was automatically stopped using the validation accuracy value, after a performance stagnation or decrease of 20 consecutive epochs. On average, LSTMs were trained on 181 ± 36 (std) epochs before achieving their best measured performance on validation data. The validation accuracy value is either computed on a validation fold of the training dataset during 5-folds cross-validation or on the test dataset during training with the whole training dataset.

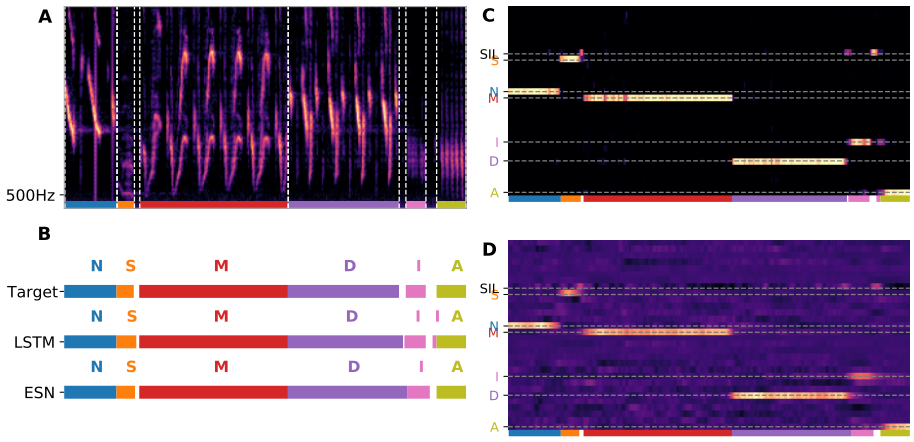

Fig. 3. An example predicted sequence of ESN and LSTM models on a song bout. **A** Mel-scale spectrogram of the song bout used as example. **B** Target annotation sequence (blank represent silence *SIL* class) and predicted annotation sequence from LSTM and ESN. Colored lines represents segment of consecutive frames sharing the same annotation. The phrase label corresponding to each segment of frames is indicated on top. Frame accuracy can be conceptualized as measuring the lines alignment in length and color between the targets and the predictions. WER only measures the validity of the colors (or phrase labels) sequence. Here, the expected sequence of phrase is *N-S-M-D-I-A*. **C** LSTM chromagram-like output activations for all frame representing the song. The y axis represents phrases classes. The x axis represents time in frames. Each frame of the song is annotated following the argmax of the activations for this frame. **D** Same as **C** with the ESN chromagram-like output activations. Because output activation of ESN is linear, the outputs values tends to be noisier than the outputs of the LSTM, that are normalized using a *softmax* activation function.

3 Results

3.1 Performance of transduction

Performance of the models on the audio-to-frame task was evaluated using a 5-fold cross-validation on the training dataset defined in section 2.2. We trained 30 random initializations of ESNs for each experiment, against 5 random initializations of LSTMs. This imbalance was motivated by the important training time of LSTMs, and by the fact that LSTMs models are expected to converge toward similar solutions independently of their initializations, as fully trained neural networks, while ESNs rely on a randomized layer of neurons kept fixed during training.

The frame accuracy (ACC) was computed for all songs, and then averaged over the whole corpus, the folds and the random instances of the models. There is no significative difference in performance between the models medians (Wilcoxon rank-sum test over all accuracy measures, for all songs and all

folds, $W = 10016240$, $p = 0.28$), and they both achieve an accuracy rate slightly higher than 93% in mean and 95% in median. We completed ACC with an F1-measure, which is significantly lower than ACC (around 0.86). This F1-measure is computed using a macro-averaging of precision and recall over all the classes of phrases, i.e. all classes are given equal importance in the computation, unlike with the accuracy score. This metric therefore gives insights on how well the models truly perform for all syllables types, without taking into account the imbalance of the classes in the dataset. All the metrics values are given in table 2. An example of models outputs can be found in figure 3.

Table 1. Average scores obtained with a 5-fold cross-validation over all training songs and several models instances.

Model	Average frame accuracy (ACC)	Median frame accuracy	F1 (macro avg.)
LSTM	0.931 ± 0.104	0.951	0.865
ESN	0.935 ± 0.09	0.952	0.877

Finally, all models instances were trained on the whole training set and evaluated on the test set defined in 2.2. ESNs achieve an accuracy rate of $94.4\% \pm 2.7\%$, while LSTMs reach a lower accuracy rate of $93.9\% \pm 10.0\%$. ESNs accuracy median (95.4%) is however significantly lower than LSTMs accuracy median (95.9%). (Wilcoxon rank-sum test over all accuracy measures on test set, $W = 534682.5$, $p = 2.92 \times 10^{-8}$).

Comparison with [4] can not be done fairly, as our method operate at phrase level and not at syllable level, and as we did not use the same dataset. We discuss the possibility of extending this work in Discussion. However, our method shows performance at frame level belonging to the same range of performance exhibited in [4] (between 92%-98% accuracy)[5].

3.2 Performance of models on reduced dataset

The aim of canary songs annotation automation is to save as much time and resources as possible for the experimenters working with these songs. Although our models are supervised, the amount of annotated data required for the models to produce acceptable results can be low enough to still significantly save time. We tried to assess the minimum amount of data necessary to reach the performances described in 3.1.

Figure 4 shows the evolution of the ACC score on the test set given the number of songs in the training dataset, using 5 instances of LSTMs and 30 instances of ESNs on random subsets of data. ESNs appear to be less sensitive to reduction of the training dataset than LSTMs, and reach back their peak performance being trained on 30 to 50 songs, which represents approximately

[5] As the work in [4] seems to not having been reviewed yet, we will not make further comparisons to avoid any mistakes than could originate from unverified results.

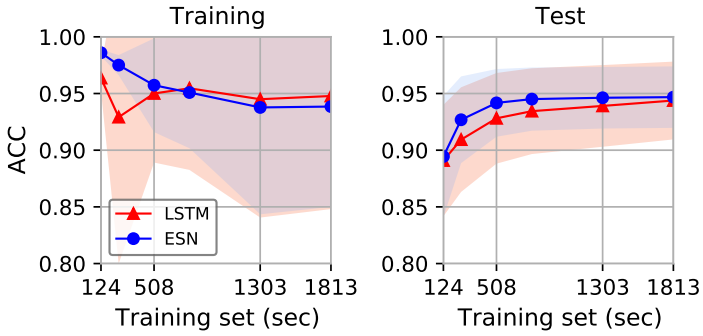

Fig. 4. Frame accuracy of models trained on subsets of songs of increasing size, on the training set (left) and on the testing set (right). Thirty songs represents around 780 seconds of song, 50 songs represents around 1303 seconds of song, 70 songs represents around 1813 seconds of song.

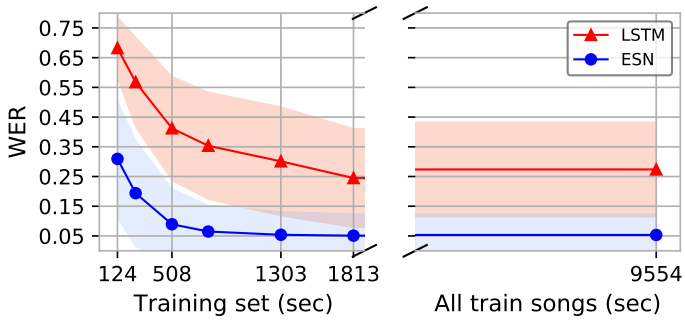

Fig. 5. WER of models trained on subsets of songs of increasing size (left) and WER of models trained on the whole training set (right). Song time is equivalent to the one explained in figure 4.

780 ± 55 to 1303 ± 87 seconds of song (13 minutes to 21 minutes 43 seconds). Low overfitting is observed with this training set sizes, as the test and train ACC are comparable. LSTMs, on the other hand, reach back their peak performance defined in 3.1 with a training set containing at least 70 songs, which represents approximately 1813 ± 87 seconds (around 30 minutes of song).

3.3 Sequence extraction

In order to perform the audio-to-phrases task, we reconstructed phrases from the models predictions made at frame level for the audio-to-frame task. To do so, we simply annotated consecutive frames sharing the same annotation with a single label. Segmentation of songs is therefore implicitely performed by the

RNN: we consider that phrases onset and offset are delimited by uninterrupted sequences of frames with the same label. Figure 3 shows that this method allows to accurately segment the songs without any post processing, because ESNs and LSTMs shift their outputs activations in time following the onset and offset of the phrases. Figure 5 shows that ESNs significantly outperform LSTMs on the audio-to-phrases task, by achieving a $5.3\% \pm 7.2\%$ WER on the testing set, being trained on the whole training set. LSTMs achieve a $27.4\% \pm 16.2\%$ WER. This difference can be explained by a higher instability of LSTMs predictions onsets and offsets, creating small noisy occurrences of phrases where they are not expected, as visible in figure 3B. LSTMs also have lower recall than ESNs for under represented syllables types such as syllable *L2* (13.0% of recall during cross-validation when using ESNs, against 4.9% of recall when using LSTMs). While these errors are insignificant at frame level, because of the important imbalance in duration between phrase classes, they can seriously hinder the WER, as it is not weighted by phrases durations. Finally, ESNs are able to reach their peak performance using a subset of 30 to 50 songs, the same number they require in 3.2 to reach their peak ACC. This allows us to recommand to use ESNs to annotate songs and to train them on at least around 20 minutes of song to obtain acceptable results.

3.4 Time constraint

All trainings where performed on an *Intel Core* i7-9850H with 12 cores operating at a frequency of $2.60GHz$, on a computer equiped with $31.1GiB$ of RAM. The training time considered takes into account the tool used to build the models – *ReservoirPy* [17] for the ESNs, *TensorFlow* and *Keras* for the LSTMs – and the policy used to stop the training, when applicable. Training times were averaged over the 10-folds cross-validation. Average training time for ESNs is significantly shorter than average training time for LSTMs, as ESNs only require one epoch of training. LSTMs on the other hand were trained using an early stopping policy based on validation accuracy with a patience of 20 epochs, and only reach top performance after around 180 epochs of training. These measures are nevertheless only given as very broad indications, as the comparison is not completely fair: ESNs were able to benefit from a parallelized training policy over the 12 cores of the processor, while the LSTMs were only partially benefiting from it. Also, most deep learning models such as LSTMs are nowadays trained on specialized hardware like GPUs. In our case, training on GPU makes the training longer, as our LSTMs are quite small, and as the flow of data between CPU and GPU adds a significant overhead. However, regarding the very short training time of ESNs, we can safely hypothesized that such performances are out of reach for an LSTM, even with further optimizations.

4 Discussion

Canary is one of the most complex singer that can be found among song birds. Its songs display sophisticated syntactic rules, as demonstrated in [12], which make

Table 2. Average training time for the two models during 10-fold cross-validation.

Model	Average training time (s)
LSTM	2930 ± 222
ESN	35 ± 1

it an appropriate candidate for deeper analysis of the emergence and learning of sequential organization of gestures. However, these analysis heavily rely on the quantity of available data to untangle the temporal complexity of the songs. These data are usually hand-annotated, and the full annotation process has to be repeated for each individual, because each canary produces specific syllables.

In this context, we proposed an efficient yet lightweight solution using RNNs to help with the annotation process, by shortening significantly the amount of time necessary to annotate large amount of data. Our method achieves a low error rate at phrase level (around 5% when using ESN) which should be acceptable for syntactic analysis. We applied this method on a canary that was producing particularly complex syllables in order to build robust classifiers applicable to any domestic canary songs. A more extensive study should be conducted soon, using other publicly available datasets. We also make ours public, to contribute to any field of research requiring annotated canary songs. Additionally, a more extensive study would also allows to perform a fair comparison with [4] by performing the annotation task at syllable level, and to assess the quality of the annotated sequence of phrases by comparing their temporal structure with the results of [12]. In any case, we confidently make the hypothesis that our method allows faster computation that the deep learning solution presented in [4], at least when using an ESN, while reaching similar performance. Indeed, the full training of an ESN only take around 40 seconds using 2 hours 39 minutes of data.

Furthermore, our method is rooted in speech recognition area by using MFCCs as representation of the audio signals, in order to extract the relevant spectral information and reduce its dimensionality. While this approach was criticized in [10], a previous study successfully using CNN and HMM to decode Bengalese finch songs, we empirically show that MFCCs seem to remain a good approximation of spectral information, at least for canary vocalizations. We also demonstrated that RNN like ESN can outperform more complicated and widely used techniques like LSTM, within a fair comparison setup, on tasks deemed as difficult and closely related to speech recognition.

As canary songs are reported to be among the most difficult vocalizations to automatically analyse, we finally make the hypothesis that our method could be successfully applied to other species, song birds, mammals, or even insects. Our method has the advantage of being easy to experiment with, as the training time for ESNs is very short compared to other deep learning algorithms, allowing for extensive optimization and statistical robustness, and does not require powerful hardware to run successfully.

Acknowledgment

We would like to thank Catherine Del Negro, Aurore Cazala and Juliette Giraudon for the recording and transcription of the canary data. We also thank Inria for the ADT fellowship grant Scikit-ESN.

References

1. Anderson, S.E., Dave, A.S., Margoliash, D.: Template-based automatic recognition of birdsong syllables from continuous recordings. J. Acoust. Soc. Am. **100**(2), 1209–1219 (1996)
2. Bay, M., Ehmann, A.F., Downie, J.S.: Evaluation of Multiple-F0 Estimation and Tracking Systems. In: ISMIR (2009)
3. Chu, W., Blumstein, D.T.: Noise robust bird song detection using syllable pattern-based hidden Markov models. In: 2011 IEEE ICASSP. pp. 345–348 (2011)
4. Cohen, Y., Nicholson, D., Gardner, T.J.: TweetyNet: A neural network that enables high-throughput, automated annotation of birdsong. bioRxiv p. 2020.08.28.272088 (2020)
5. Gallicchio, C., Micheli, A., Pedrelli, L.: Fast Spectral Radius Initialization for Recurrent Neural Networks. In: Recent Advances in Big Data and Deep Learning. pp. 380–390 (2020)
6. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: Continual prediction with LSTM. In: ICANN 1999. vol. 2, pp. 850–855 (1999)
7. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks-with an erratum note'. German National Research Center for Information Technology GMD Technical Report **148** (2001)
8. Juang, B.H., Rabiner, L., Wilpon, J.: On the use of bandpass liftering in speech recognition. IEEE Trans. Acoust. Speech Signal Process. **35**(7), 947–954 (1987)
9. Kaewtip, K., Alwan, A., O'Reilly, C., Taylor, C.E.: A robust automatic birdsong phrase classification: A template-based approach. J. Acoust. Soc. Am. **140**(5), 3691–3701 (2016)
10. Koumura, T., Okanoya, K.: Automatic Recognition of Element Classes and Boundaries in the Birdsong with Variable Sequences. PLOS ONE **11**(7), e0159188 (2016)
11. Leitner, S., Catchpole, C.K.: Syllable repertoire and the size of the song control system in captive canaries (Serinus canaria). J. Neurobiol. **60**(1), 21–27 (2004)
12. Markowitz, J.E., Ivie, E., Kligler, L., Gardner, T.J.: Long-range Order in Canary Song. PLOS Computational Biology **9**(5), e1003052 (2013)
13. McFee, B., Lostanlen, V., Metsai, A., McVicar, M., Balke, S., Thomé, C., Raffel, C., Zalkow, F., Malek, A., Dana, Lee, K., Nieto, O., Mason, J., Ellis, D., Battenberg, E., Seyfarth, S., Yamamoto, R., Choi, K., viktorandreevichmorozov, Moore, J., Bittner, R., Hidaka, S., Wei, Z., nullmightybofo, Hereñú, D., Fabian-Robert Stöter, Friesch, P., Weiss, A., Vollrath, M., Kim, T.: Librosa/librosa: 0.8.0. Zenodo (2020)
14. Sainburg, T., Thielk, M., Gentner, T.Q.: Latent space visualization, characterization, and generation of diverse vocal communication signals. bioRxiv p. 870311 (2020)
15. Tachibana, R.O., Oosugi, N., Okanoya, K.: Semi-Automatic Classification of Birdsong Elements Using a Linear Support Vector Machine. PLOS ONE **9**(3), e92584 (2014)

16. Tan, L.N., Alwan, A., Kossan, G., Cody, M.L., Taylor, C.E.: Dynamic time warping and sparse representation classification for birdsong phrase classification using limited training data. J. Acoust. Soc. Am. **137**(3), 1069–1080 (2015)
17. Trouvain, N., Pedrelli, L., Dinh, T.T., Hinaut, X.: ReservoirPy: An Efficient and User-Friendly Library to Design Echo State Networks. In: Artificial Neural Networks and Machine Learning – ICANN 2020. pp. 494–505 (2020)
18. Waser, M.S., Marler, P.: Song learning in canaries. J. Comp. Physiol. Psychol. **91**(1), 1–7 (1977)