Hierarchical-Task Reservoir for Anytime POS Tagging from Continuous Speech

Luca Pedrelli, Xavier Hinaut

Hierarchical-Task Reservoir for Anytime POS Tagging from Continuous Speech

Luca Pedrelli
1. INRIA Bordeaux Sud-Ouest.
2. LaBRI, Bordeaux INP, CNRS, UMR 5800.
3. Institut des Maladies Neurodégénératives,
Université de Bordeaux, CNRS, UMR 5293.
Bordeaux, France.
orcid.org/0000-0002-4752-7622

Xavier Hinaut
1. INRIA Bordeaux Sud-Ouest.
2. LaBRI, Bordeaux INP, CNRS, UMR 5800.
3. Institut des Maladies Neurodégénératives,
Université de Bordeaux, CNRS, UMR 5293.
Bordeaux, France.
orcid.org/0000-0002-1924-1184

***Abstract*—We propose a novel architecture called Hierarchical-Task Reservoir (HTR) suitable for real-time sentence parsing from continuous speech. Accordingly, we introduce a novel task that consists in performing anytime Part-of-Speech (POS) tagging from continuous speech. This HTR architecture is designed to address three sub-tasks (phone, word and POS tag estimation) with increasing levels of abstraction. These tasks are performed by the consecutive layers of the HTR architecture. Interestingly, the qualitative results show that the learning of sub-tasks enforces low frequency dynamics (i.e. with longer timescales) in the more abstract layers. We compared HTR with a baseline hierarchical reservoir architecture (in which each layer is an ESN that addresses the same POS tag estimation). Moreover, we also performed a thorough experimental comparison with several architectural variants. Finally, the HTR obtained the best performance in all experimental comparisons. Overall, the proposed approach will be a useful tool for further studies regarding both the modeling of language comprehension in a neuroscience context and for real-time implementations in Human-Robot Interaction (HRI) context.**

***Index Terms*—Recurrent Neural Networks, Hierarchical Reservoir Computing, Natural Language Processing, Speech Recognition, Part-of-Speech, POS tagging, Anytime Process, Hierarchical Processing.**

I. Introduction

Language unfolds in time. Both the brains of the speaker and the hearer need to produce/process complex acoustic streams. We easily forget the complexity at hand when we speak our native language. However, when one hears an unknown foreign language, little or no information could be extracted from the acoustic stream. Another crucial property of spoken language is the short availability of the signal (compared to written text): the brain needs to process it as quickly as possible because the phonological memory buffer is very limited [1].

Natural Language Processing (NLP) tools (e.g., CoreNLP, NLTK, Gensim and SpaCy) are quite different from how our brains work. One example is the typical use of bag-of-words (i.e. considering a sentence as unordered words instead of considering the timing of the sequence of words). Most of the recent NLP tools based on deep learning approaches [2]–[4] are focused on encoder/decoder mechanisms and bidirectional architectures to address temporal and more complex dependencies between words. However, these implementations need to parse the whole sentence before producing an output. On the contrary, brains process sentences online in an *anytime* fashion (i.e. ability to have a partial understanding before the end of the sentence).

For instance, in the case of human-robot interaction through spoken language [5]–[7], a common approach is still to perform parsing of sentences based on a restricted grammar parser written by hand, with speech preprocessing done with Google Speech API. It seems that even with Deep Learning based architecture that try to integrate NLP and vision processing, there is still the need for ad-hoc word correction from the speech API [8]. Overall, these steps are performed by different modules instead of being a fully integrated architecture. Typically the speech recognition module is implemented through a deep learning approach based on sequence transduction approaches [9]–[11]. In these approaches there are two main limitations: 1) one need to know in advance the maximal length of the sentences in the test phase and 2) the speech module needs to read the whole sentence in order to produce the phone recognition of the input signal. Therefore, these approaches are not suitable for a real-time human-robot interaction system or for biologically plausible cognitive models.

Since another important characteristic of the brain is the presence of hierarchical structures the deep learning (DL) paradigm took inspiration from this fact: for example for the vision areas hierarchy (cortical areas V1, V2, V3, V4, etc.). However, this is only a *shallow* inspiration, because the brain is processing information in a much more dynamic way: exemplified for instance by the presence of *feedforward* and *feedback* connections [12] (e.g. there are strong feedback connections from area V4 to area V2). In particular, deep Recurrent Neural Networks (RNNs) [13]–[15] are a class of neural networks suitable for time series processing able to intrinsically develop hierarchical and distributed temporal features [16], [17]. However, from the learning mechanisms point of view the use of back-propagation in deep neural networks makes these approaches not biologically plausible.

A good solution based on RNNs that avoids the use of back

propagation is the Reservoir Computing (RC) paradigm [18], [19]. Indeed, RC is widely used to model brain processes with RC instances such as Echo State Networks (ESNs) and Liquid State Machines [19]. In this case, we focus on the ESN model that represents a discrete-time system.

More recent works introduce hierarchically organized models [9], [20] and deep recurrent architectures [16], [17] in the context of RC. In particular, Hierarchical Reservoir Computing (HRC) obtained results competitive with the state-of-the-art on continuous speech recognition [9].

Following these aspects, the aim of this paper is to introduce a novel model called Hierarchical-Task Reservoir (HTR) for anytime sentence parsing from continuous speech which combines the following aspects: i) suitable for real-time implementations of sentence parsing from continuous speech, ii) learning different levels of abstraction through a hierarchy of sub-tasks, iii) ability to learn a hierarchical representation of temporal features, iv) a suitable tool for neuroscience linguistic studies.

Based on such considerations, we construct a novel sentence parsing dataset for Anytime POS tagging basing on the TIMIT continuous speech recognition corpus [21]. TIMIT is one of the most used dataset for studies and analyses on continuous speech recognition in the context of neural networks [9], [10]. With the SpaCy library we computed the POS labels starting from the sentences of the corpus. We quantitatively and qualitatively compare HTR and HRC architectures on the Anytime POS task. Then, we qualitatively study the dynamics developed in the layers of the HTR architecture. Finally, we quantitatively compare architectural variants of HTR to empirically study the effect of the hierarchy of sub-tasks on the model performance.

II. Reservoir Computing

Reservoir Computing (RC) [22] is a paradigm for the design and training of Recurrent Neural Networks, in which the recurrent layer is non-linear, randomly initialized and left untrained. Within RC networks, in this work we are focused on Leaky Integrator Echo State Networks (LI-ESNs) [23]. As shown in Figure 1, the architecture of LI-ESN is composed by a non-linear recurrent layer called the *reservoir* and a linear output layer called *readout*. Omitting the bias in the following formulas for the ease of notation, the state transition function of the LI-ESN is computed as follows:

$$\mathbf{x}(t) = (1-a)\mathbf{x}(t-1) + a\,\mathbf{tanh}(\mathbf{W}_{\text{in}}\mathbf{u}(t) + \mathbf{W}\mathbf{x}(t-1)), \quad (1)$$

where $\mathbf{u}(t) \in \mathbb{R}^{N_U}$ is the input at time t, $\mathbf{x}(t) \in \mathbb{R}^{N_R}$ is the reservoir state, $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_R \times N_U}$ is the input matrix, $\mathbf{W} \in \mathbb{R}^{N_R \times N_R}$ is the recurrent matrix, $a \in [0, 1]$ is the leaky parameter and $\mathbf{tanh}$ is the element-wise hyperbolic tangent. The reservoir is initialized considering the echo state property [24]. Accordingly, the values of matrix $\mathbf{W}$ are randomly initialized from a uniform distribution and then rescaled in order to have a spectral radius ρ (i.e., the maximum absolute eigenvalue of $\mathbf{W}$) less than 1. However, in practical cases the ESP can be satisfied also with values of $\rho \geq 1$ [25]. The values in matrix $\mathbf{W}_{\text{in}}$ are randomly initialized from a uniform distribution and then rescaled in order to have a norm σ. The output of LI-ESN is computed as follows:

$$\mathbf{y}(t) = \mathbf{W}_{\text{out}}\mathbf{x}(t), \quad (2)$$

where $\mathbf{y}(t) \in \mathbb{R}^{N_Y}$ is the output at time t and $\mathbf{W}_{\text{out}}$ is the output matrix. The values in $\mathbf{W}_{\text{out}}$ represent the free-parameters of the model which are trained to solve the task by computing typical approaches for linear optimization such as ridge regression. In the following, we use the term *ESN* to refer to LI-ESN model.

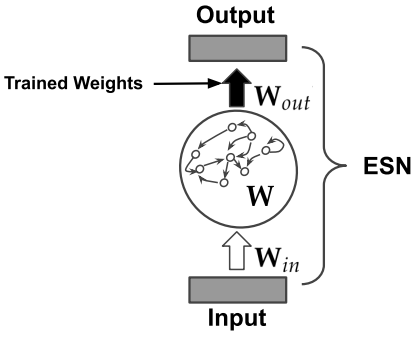

Fig. 1. **An example of Echo State Network architecture.**

Hierarchical Reservoir Computing (HRC) [9] is a RC approach that obtained promising results in Acoustic Modeling. HRC networks are composed by a cascade of ESNs in which each network is individually trained on the basis of the output of the previous ESN. In this way the higher models can correct the errors produced by the previous models. As shown by recent studies regarding Deep Reservoir Computing [16], [17], a main advantage offered by the layering in recurrent architectures is the ability to develop a hierarchical organization of temporal features independently by the learning algorithm.

III. Hierarchical-Task Reservoir

The source code of the experiments is available on our Github repository: https://github.com/lucapedrelli/PedrelliHinaut2020_IJCNN

In this paper, we introduce a novel architecture based on HRC that we call Hierarchical-Task Reservoir (HTR). The architecture is composed by a hierarchy of layers in which each layer is an ESN optimized on a different task. Each ESN is fed by the previous ESN in the hierarchy and the pipeline is progressively optimized from the first layer to the last layer. The main differences between HTR and the approach proposed in [9] are two: (i) instead of addressing only phone recognition tasks, HTR architectures deal with a different kind of task for each layer with a progressively increased level of abstraction (corresponding to a decreased frequency of labels). In this way, the learning process influences progressively the level of abstraction of the representation in higher layers; (ii) instead of controlling by hand the internal dynamics of the higher layers with fixed input scales and fixed leaky integrators, in HTR we optimize all the hyper-parameters in each layer. In

this way, the hyper-parameters are optimized depending on the task addressed by the layer.

The Figure 2 shows an example of the maximal size HTR architecture used in this paper. The state computation of the HTR architecture for the layer $l = 1, ..., N_L$ is defined by following equations:

$$\mathbf{x}^{(l)}(t) = (1 - a^{(l)})\mathbf{x}^{(l)}(t-1) + a^{(l)}\,\mathbf{f}(\mathbf{W}_{\text{in}}^{(l)}\mathbf{i}^{(l)}(t) + \mathbf{W}^{(l)}\mathbf{x}^{(l)}(t-1)) \tag{3}$$

in which the input $\mathbf{i}^{(l)}$ of layer l is defined as follow:

$$\mathbf{i}^{(l)}(t) = \begin{cases} \mathbf{u}(t) & \text{if } l = 1 \\ \mathbf{y}^{(l-1)}(t) & \text{if } l > 1. \end{cases} \tag{4}$$

where $\mathbf{u}^{(1)} \in \mathbb{R}^{N_U^{(1)}}$ is the external input with an input dimension of $N_U^{(1)}$ and $\mathbf{y}^{(l-1)}(t) \in \mathbb{R}^{N_U^{(l)}}$ is the output of layer $l-1$ where $N_U^{(l)}$ is the dimension of the output of layer $l-1$ which coincides with the dimension of the input of layer l. The output of layer l is computed as follows:

$$\mathbf{y}^{(l)}(t) = \mathbf{W}_{\text{out}}^{(l)}\mathbf{x}^{(l)}(t), \tag{5}$$

where $\mathbf{x}^{(l)}(t) \in \mathbb{R}^{N_R}$ is the reservoir state of layer l, $\mathbf{W}_{\text{in}}^{(l)} \in \mathbb{R}^{N_R \times N_U^{(l)}}$ is the input matrix of layer l with a input dimension of $N_U^{(l)}$, $\mathbf{W}^{(l)} \in \mathbb{R}^{N_R \times N_R}$ is the recurrent matrix of layer l, $\mathbf{W}_{\text{out}}^{(l)}$ is the output matrix, $a^{(l)} \in [0, 1]$ is the leaky parameter and $\mathbf{f}^{(l)}$ is the activation function of layer l. In this work we consider $\mathbf{f}^{(l)} = \mathbf{tanh}$ for each l. The values of matrices $\mathbf{W}_{\text{in}}^{(l)}$ and $\mathbf{W}^{(l)}$ are randomly initialized from uniform distribution and then rescaled such that the euclidean norm of $\mathbf{W}_{\text{in}}^{(l)}$ is equal to $\sigma^{(l)}$ and the spectral radius of $\mathbf{W}^{(l)}$ is equal to $\rho^{(l)}$. It deserves to be mentioned that the values $\rho^{(l)}$ and $a^{(l)}$ have an important role in controlling stability of deep recurrent neural networks [26].

In HTR architecture, each ESN is individually optimized on a different task by performing random search (for hyper-parameter tuning) and ridge regression (for readout learning) following the pipeline from the layer 1 to the layer N_L. The main architecture that we consider is composed by 3 layers: (i) the first layer is optimized to estimate phones from speech SP→PH, (ii) the second layer is optimized to estimate words from phones PH→WD and (iii) the third layer is optimized to estimate POS tagging from words WD→POS. Overall, we denote the whole HTR architecture as SP→PH→WD→POS. The Algorithm 1 describes the optimization procedure for HTR considering N_L layers, N_{Trials} trials and external input $\mathbf{i}^{(1)}$. In the following section we provide the description of the considered tasks.

Algorithm 1 HTR Training

procedure TRAINHTR($\mathbf{i}^{(1)}$, N_{Trials})
  for `l in 1, ...,` N_L **do**
    $\theta^{(l)} = \texttt{generateTrials}(N_{Trials})$
      ▷ generate N_{Trials} models
    $\texttt{HTR}^{(l)} = \texttt{modelSelection}(\theta^{(l)}, l)$
      ▷ select the best model on task l
    $\mathbf{i}^{(l+1)} = \texttt{computeOutput}(\texttt{HTR}^{(l)}, \mathbf{i}^{(l)})$
      ▷ compute the output as input for the next layer
  return HTR
    ▷ return the whole trained architecture

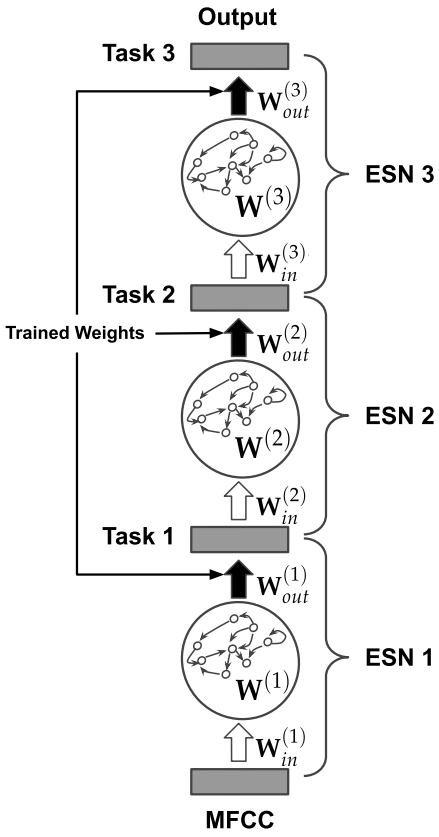

Fig. 2. **HTR architecture: the main model we propose to address the hierarchical task SP→PH→WD →POS.** The architecture receive the features extracted from the speech signal with MFCC algorithm each 10 ms. The first layer is optimized for SP→PH (**Task 1**), the second layer is optimized for PH→WD (**Task 2**) and, finally, the third layer is optimized for WD→POS (**Task 3**).

IV. Anytime POS Tagging from Speech Recognition

In this section, we introduce a novel grammatical analysis task for anytime part-of-speech (POS) tagging from continuous speech recognition. It consists in a supervised classification task in which the aim is to continuously provide the POS Tagging estimation for each 10 ms frame of speech signal. Accordingly, the approaches evaluated on this task must be suitable for real-time implementations.

For this purposes, we focus on the TIMIT [21] corpus which is widely used in the field of continuous speech recognition. The dataset contains recordings from 630 American speakers and each speaker read 10 phonetically rich sentences. Each frame of 10 ms is labelled with a phone and a word class with a total of 61 phones and 6012 words. We divided the dataset in a training set containing 540 speaker and a test set containing 90 speaker. We use the last 135 speaker for the validation set.

As in [9], we reduce the number of phones from 61 to 51 merging similar sounds. In order to keep efficiency in training processes, we considered only the 50 most frequent words of the corpus, each having one output label, and merging the others in a single out-of-vocabolary (OOV) label.

We extended the dataset by computing the POS tagging by using *SpaCy* tools for each sentence with a total of 17 grammatical elements. The POS labelling is performed by computing a POS tag for each word. Then, the POS tag is associated with a class label for each frame that belongs to the duration of the word in the speech signal. Finally, a silence label is considered as an additional class in correspondence with silence frames in the speech signal. For each 10ms the Mel Frequency Cepstral Coefficients (MFCC) [9] algorithm computes the 39 components used as input for the architectures (see [9] for the setup of MFCC).

In order to form the whole hierarchical task for POS tagging and to define the different architectures for the experiments, we consider 10 sub-tasks listed in the following: SP→PH (predict phones from speech), PH→WD (words from phones), WD→POS (POS from words), SP→WD (words from speech), SP→POS (POS from speech), PH→POS (POS from phones), WD→POS (POS from words), WD(lb)→POS (POS from words labels), PH(lb)→WD (words from phones labels) and PH(lb)→POS (POS from phones labels). In all tasks, the evaluation is performed by computing the frame error rate (FER) which is the ratio between the number of correctly classified time steps and the total number of time steps.

For each task, the model selection is performed with a random search evaluating the performance on the validation set. For each trial, we sample the hyper-parameters spectral radius in the following way: $\rho^{(l)}$ and input norm $\sigma^{(l)}$ from a logarithmic distribution in $[0.1, 10]$, leak-rate $a^{(l)}$ from a uniform distribution in $[0.1, 1]$ and ridge (regularisation) $\lambda^{(l)}$ from a discrete uniform distribution in $[10^0, 10^{-1}, ..., 10^{-7}, 10^{-8}]$. In training phase a search grid is performed on $\lambda^{(l)}$ considering values reported above. As in standard RC, each hyper-parametrization (i.e. $\rho^{(l)}$, $\sigma^{(l)}$, $a^{(l)}$ and $\lambda^{(l)}$) is evaluated averaging the results obtained by randomly initialized architectures (that we call *guesses*). In this work we consider 5 guesses.

In order to keep the execution of the hyper-parameter search in a reasonable time, we did not optimize the input scaling of each MFCC input component individually. Triefenbach et al. [9] optimized inputs scaling by clusters: optimizing the scaling for signals, signal derivative (delta), and signal acceleration (i.e. delta-delta). Moreover, since we have several experiments to perform (due to a thorough architectural comparison) on a significantly big corpus we performed preliminary experiments to fix a proper number of units. We find that 1000 units are a good tradeoff between accuracy and efficiency, considering also that the proposed approach must be efficiently executable in real-time. Accordingly, in the following all experiments are performed by using $N_R = 1000$ for each ESN layer.

The 3 main architectures that we consider in this paper are listed in Table I. Therefore, in HTR first the layer ESN 1 (see Figure 2) is optimized to estimate phones (PH) from speech (SP) then ESN 2 is optimized to recognize words (WD) from phones and finally ESN 3 is optimized to estimate POS tagging from words. Differently, in HRC case, ESN 1 is optimized to estimate POS from HRC while ESN 2 and ESN 3 are both optimized to estimate POS tagging from POS tagging. The ESN in I is optimize to estimate POS tagging from speech implementing 3000 units (the same total number of units of HTR and HRC).

TABLE I
THE TASK STRUCTURE OF THE ARCHITECTURES HTR, HRC AND ESN.

Name	Architecture
HTR	**SP→PH →WD→POS**
HRC	SP→POS→POS→POS
ESN	SP→POS

V. EXPERIMENTAL RESULTS

A. Architecture Comparison on the Anytime POS Task

Here, we present the results obtained by HTR, HRC and ESN on the Anytime POS Task. Table II shows the test FERs achieved by HTR, HRC and ESN. The HTR obtained the best result with a test FER of 45.51% followed by HRC and ESN that achieved 49.55% and 53.40% test FER, respectively. First, we can note that HRC achieve 4.08 FER points less than ESN. This highlights that solving the Anytime POS task with a pipeline of ESN modules, as performed in HRC, allows us to improve the results with respect to the use of a single ESN module (containing a total equivalent number of reservoir units). Moreover, the hierarchical-task approach implemented by HTR (i.e. considering progressively more abstract sub-tasks in higher layers) allows us to have a further improvement on the Anytime POS task outperforming HRC architecture by 4.04 FER points.

TABLE II
TEST FERS: HTR, HRC AND ESN ON THE ANYTIME POS TASK.

Architecture	Test
HTR	$\mathbf{45.51(0.04)\%}$
HRC	$49.32(0.12)\%$
ESN	$53.40(0.15)\%$

B. Improving HRC Perf. by increasing the number of layers

In this section, we show the experimental results obtained by HRC on the Anytime POS Task by varying the number of layers. Table III shows the test FERs achieved by HRC considering progressively more layers in the architecture (i.e., 1, 2 and 3 layers). The HRC architecture obtained 50.35%, 49.55% and 49.32% test FER for 1, 2 and 3 layers, respectively. We can note that the error obtained by the HRC architecture decreases when the number of layers increases. This confirms the ability of the next layers to correct the errors made by the previous layers as achieved in [9]. This aspect is also interesting in relation to results obtained by the ESN with 3000 units (see Table II). This suggests that, in terms of

TABLE III
TEST FERS OBTAINED ON THE ANYTIME POS TASK BY PROGRESSIVELY ADDING LAYERS IN THE HRC ARCHITECTURE.

HRC	FER
SP→POS	$50.35(0.19)\%$
SP→POS→POS	$49.55(0.14)\%$
SP→POS→POS→POS	$\mathbf{49.32(0.12)}\%$

performance on the considered task, a hierarchical pipeline of ESN moules is much more effective than a big single ESN.

C. Improving HTR Performance with the Hierarchical Task

In this section, we show the experimental results obtained by HTR on the Anytime POS Task by varying the number of layers. Table IV shows the test FERs achieved by HTR considering progressively more layers in the architecture (i.e., 1, 2 and 3 layers). The HTR architecture progressively improve the performance when the number of layers increases obtaining 50.35%, 47.62% and 45.51% test FER for 1, 2 and 3 layers respectively. Figure 3 shows the test FERs obtained by HTR and HRC architecture with 1, 2 and 3 layers. First,

TABLE IV
TEST FERS OBTAINED ON THE ANYTIME POS TASK BY PROGRESSIVELY ADDING LAYERS IN THE HTR ARCHITECTURE.

HTR	FER
SP→POS	$50.35(0.19)\%$
SP→PH→POS	$47.62(0.12)\%$
SP→PH→WD→POS	$\mathbf{45.51(0.04)}\%$

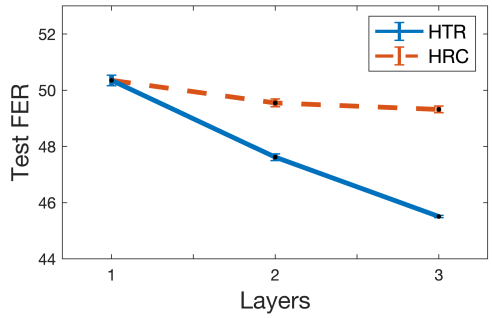

Fig. 3. **Test FER obtained by HTR and HRC when the number of layers increases.** The vertical intervals represent the standard deviation achieved by the models.

we can note that the ability of HTR to correct errors among the layers is better than the HRC case. Interestingly, while HTR obtained a significant improvement when the number of layers increases, the performance tends to saturate quickly in the case of HRC. This highlights the importance of the HTR architecture which addresses sub-tasks with an increasing level of abstraction.

D. Qualitative Comparison Between HTR and HRC

In this section, we show a qualitative comparison between HTR and HRC of the Anytime POS estimation when the number of layers increases. The sentence considered is "don't ask me to carry an oily rag like that" with the associated the POS tags "AUX VERB PRON PART VERB DET ADJ NOUN SCONJ DET". Figure 4a represents the 39 components computed by the MFCC algorithm over the input audio at each time step of 10ms. Figure 4b shows the output values of layer 1, Figure 4c shows the output values of layer 2 and Figure 4d shows the output values of layer 3.

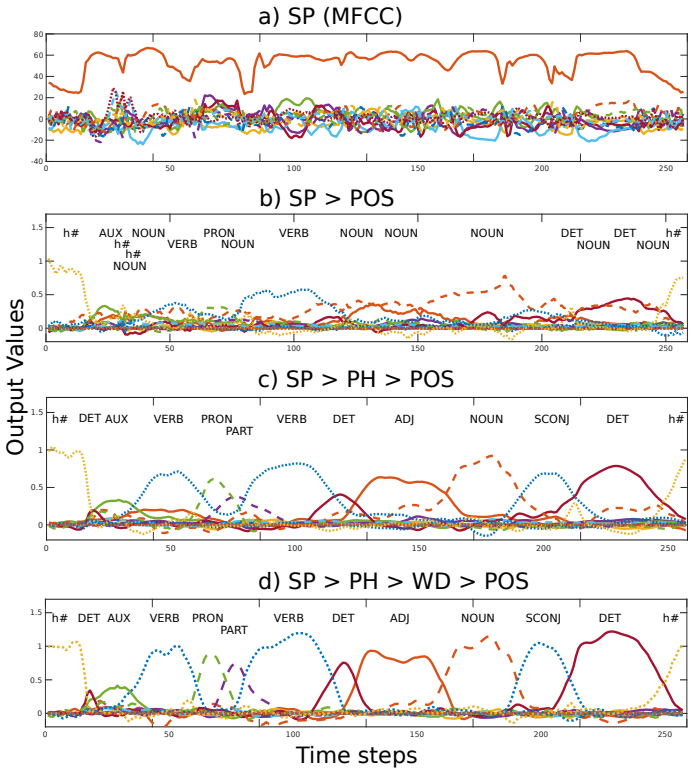

Fig. 4. **Input and outputs for the full HTR architecture (3 layers).** The Figure a) shows the components of the MFCC computed from the input audio relative to a pronounced sentence. Figures b), c) and d) show the output values of the HTR architecture with 1, 2 and 3 layers respectively. The x-axis represents the time and the y-axis represents the values. At several time points, the label corresponding to the output with the maximum activation is indicated.

In Figures 4b, c and d, each line represents the value of an output neuron over the time, each neuron is associated to a POS class and the output of the model at each time step is the POS class associated to the neuron with the maximum value. The estimated POS tags are shown in Figures 4 above the output values. From Figure 4b we can see that in many areas the estimation is not precise since some output units have overlapped ranges. For instance after the estimation of silence (label #h) in the firsts time steps the model suffers from uncertainty with a consequent decrease of accuracy for those predictions. Interestingly, from Figure 4c we can see that considering one more layer in the HTR architecture the estimation is significantly improved since the predictions are more clearly separated. This means that performing the phone recognition before the POS estimation improves significantly the quality of the classifier. Finally, in the HTR with 3 layers

the (see Figure 4d) the quality of the classification is further improved.

Figures 5 show the qualitative analysis performed on the top layers of the HRCs with 1, 2 and 3 layers. Although between

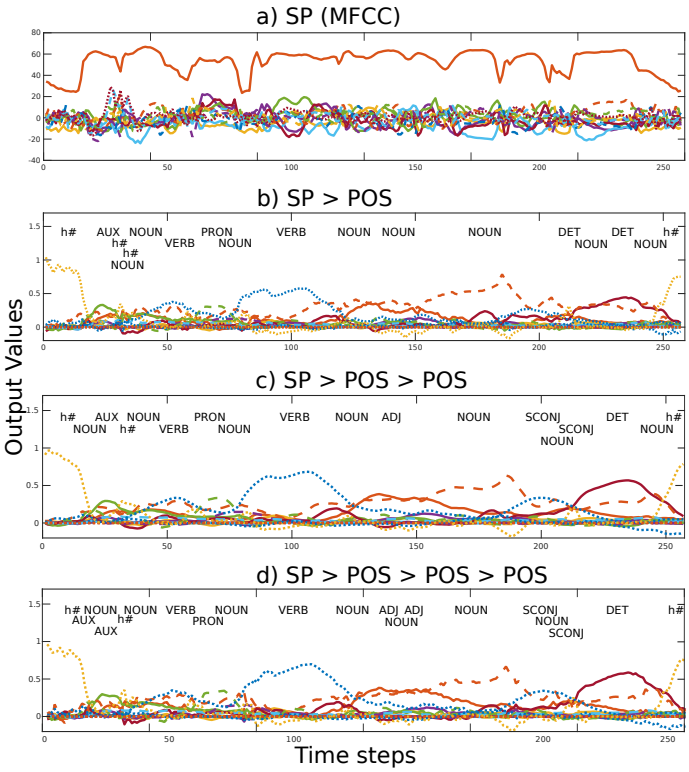

Fig. 5. **Input and outputs for various HRC architectures (layers 1, 2 and 3).** The Figure a) shows the components of the MFCC computed from the input audio relative to a pronounced sentence. Figures b), c) and d) show the output values of the HRC architecture with 1, 2 and 3 layers respectively. The x-axis represents the time and the y-axis represents the values. At several time points, the label corresponding to the output with the maximum activation is indicated.

the first and the second layers we have a small improvement of predictions if we compare the values of the output layers of HRC and HTR we can note that HTR provides a significantly better quality of classification (see Figure 5d and Figure 4d).

E. Qualitative Analysis of HRT

In this section, we present the qualitative analysis performed on the HTR architecture with 3 layers (SP→PH→WD→POS). Figures 6b, c and d show the predictions of the layer 1, 2 and 3 relative to the phone, words and POS classifications respectively. As we expected, since tasks are different and progressively more abstract, we can see from Figures 6b and 6c that the layers values goes from an higher to a lower frequency of prediction. This means that the readouts trained on different kinds of tasks with different label frequencies force progressively the dynamic of the layers from high to low frequencies.

In Figures 6c and d, at the dashed vertical lines, we can see an interesting example in which the model is able to estimate the correct POS even if the word is OOV (i.e. label class for

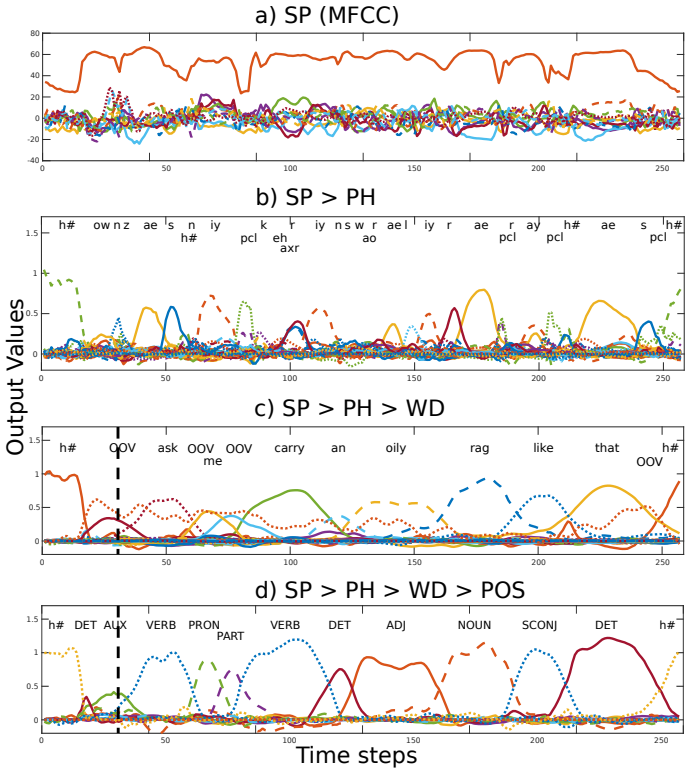

Fig. 6. **Input and outputs of the intermediate outputs of the full HTR architecture (3 layers).** The Figure a) shows the components of the MFCC computed from the input audio relative to a pronounced sentence. Figures b), c) and d) show the output values of the the layers 1, 2 and 3 of the HTR architecture (i.e. ESN 1, ESN 2 and ESN 3 in Figure 6). The x-axis represents the time and the y-axis represents the values. At several time points, the label corresponding to the output with the maximum activation is indicated.

words not in TOP50 most frequent words). In this example, the ground through labels are "*don't*" for the WD task and "*AUX*" (auxiliary verb) for the POS task. This highlights that although the last layer is fed without the direct information of the previous word the carried information is rich enough to perform a correct classification. It is also worth to mention that also in the case of readouts progressively trained in this qualitative evidence highlights the ability of a layer trained in a different task

F. Architectural Variants of HRT

In this section, we present the results of other architectural variants of HTR. First, we compare SP→PH→POS with SP→WD→POS in order to evaluate a variant of HTR without the phone recognition task. Table V shows the test errors obtained by the two considered architectural variants. Interestingly, SP→WD→POS is outperformed by both SP→PH→POS and SP→PH→WD→POS (see Table IV). This highlights that it is important to address the task SP→PH for layer 1 instead of addressing directly the task SP→WD.

Moreover, the results presented in Table IV suggest that the 3^{rd} layer (related to the task WD→POS) plays a relevant role in the achievement of the global task. Indeed, the error decreased from 47.62% to 45.51% FER. This highlights the

TABLE V
TEST FERS ACHIEVED BY ARCHITECTURES SP→PH→POS AND SP→WD→POS ON THE ANYTIME POS TASK.

Architectures	FER
SP→PH→POS	$\mathbf{47.62(0.12)\%}$
SP→WD→POS	$48.52(0.32)\%$

importance of the 3^{rd} layer in the correction of errors made by the previous layer. Then the obtained results highlight that SP→PH→WD→POS is the best choice to improve the performance on the Anytime POS task.

However, the use of only 50 words out of 6012 for the WD→POS task could be a bottleneck for the richness of the information used to solve the POS task. Therefore, a skip connection between layer 1 and layer 3 could carry more information and accordingly improve the results. The skip connection is obtained by concatenating the input of ESN 3 with the output of ESN 1 (see Figure 2), we call this variant HTR-skip. Table VI shows the test errors achieved by HTR and HTR-skip. We can see from Table VI that HTR-skip and

TABLE VI
TEST FERS ACHIEVED BY HTR AND HTR-SKIP ON ANYTIME POS TASK.

Architectures	FER
HTR	$45.511(0.04)\%$
HTR-skip	$45.506(0.04)\%$

HTR obtained very similar results. The lack of performance improvement and the qualitative analysis shown in Figure 6 suggest that the ESN 2 (see Figure 2) can carry enough rich information even using only the estimation of the 50 words + OOV label.

Overall, the results obtained by HRC in Table III (i.e. without intermediate tasks) and the results obtained by architectural variants in Table V (i.e. with only SP→PH or SP→WD intermediate task) suggest that using several intermediate tasks is crucial to obtain the best performance.

G. Ground Truth Architectures

Here, we present the results obtained by architectural variants which address the Anytime POS task starting from the ground truth of phones or words instead of starting from the speech signal. In this way, these architectures represent the performance that the model can reach on the Anytime POS task with the help of a perfect phone recognition system (i.e. PH(lb)) or a perfect word recognition system (i.e. WD(lb)). Table VII shows the test FERs obtained by WD(lb)→POS, PH(lb)→WD→POS and PH(lb)→POS where WD(lb) and PH(lb) are the ground truth of words and phones respectively. As expected WD(lb)→POS obtained the best performance with a FER of 30.03% followed by PH(lb)→WD→POS with 41.40% FER and PH(lb)→POS with 41.81% FER.

H. Comparing Results

Comparing results from TableVII and Table IV it seems counter-intuitive that adding the WD task in-between the

TABLE VII
TEST FERS ACHIEVED BY WD(LB)→POS, PH(LB)→WD→POS AND PH(LB)→POS ARCHITECTURES ON ANYTIME POS TASK.

Architectures	FER
WD(lb)→POS	$\mathbf{30.03(0.36)\%}$
PH(lb)→WD→POS	$41.40(0.13)\%$
PH(lb)→POS	$41.81(0.31)\%$

PH→POS flow improves performance mostly when starting from the speech signal (from 47.62% to 45.51%) and not when starting from the ground truth phones (from 41.81% to 41.81%). Therefore, it suggests that the WD task is actually used to correct errors made at the phone-level (ESN 1) in the full HTR architecture (i.e. SP→PH→WD→POS). This is another aspect that confirms the usefulness of our Hierarchical-Task approach, since multiple tasks with an increasing level of abstraction enable the correction of previous errors.

Moreover, we can say that the performance achieved by WD(lb)→POS architecture (30.03% FER) shows that obtaining POS labels starting from phones (PH) significantly increases the difficulty of the task compared to starting from words (WD). Therefore, this suggests that using the WD task in the full HTR architecture is crucial to have better performances on POS tagging.

Finally, it is worth mentioning that the baseline performance that we could expect to reach with the full HTR architecture (41.40% FER with PH(lb)→WD→POS) starting from the speech signal, is not so far away from the 45.51% obtained starting from a perfect phone recognition system PH(lb).

VI. DISCUSSION

In this work, with the HTR architecture we introduced a novel tool suitable for real-time sentence parsing [27] of continuous speech. For this purpose, we introduced a novel task for the Anytime POS tagging from continuous speech extending the TIMIT corpus. The HTR is composed by a hierarchy of ESNs in which each ESN is optimized on a different task (i.e. phone, word, POS tag recognition). The level of abstraction of the task increases when the layer's level increases.

The qualitative experiments performed on the output values of the layers highlight that the learning of sub-tasks with a progressively decreasing temporal frequency of labels for higher layers force a progressively low temporal frequency dynamic according to the depth of the architecture. This ability can help the HTR to progressively develop a proper level of abstraction of the input signal in order to improve the performance of the whole task.

Moreover, we quantitatively compared HTR with a typical hierarchical reservoir architecture that implements the estimation of POS tags in each layer. Then, we also compare the HTR with several architectural variants. The quantitative results show that the whole hierarchy of sub-tasks implemented in HTR is crucial to significantly improve the performance. Finally, HTR obtain only about 4 points of FER less than the

architecture trained starting from the outputs of a perfect phone recognition system (PH(lb)→WD→POS). We can conclude that HTR is competitive with the state-of-the-art approaches in the RC field on the Anytime POS task.

Overall, the HTR model and the Anytime POS task, introduced in this paper, are interesting tools for further studies regarding the language comprehension in neuroscience approaches [28], [29] or for the implementation of a real-time human-robot interaction (HRI) [5]–[7], [30].

In future work, it would be interesting to get inspiration from neurobiological findings on brain hierarchy: in primate brains there are *feedforward* and *feedback* connections between brain areas of different abstraction levels [12]. Indeed, information does not only go from sensory (i.e. less abstract) to more integrated areas (i.e. more abstract), it also flows from more abstract to less abstract areas. Thus, hierarchical models could be designed to incorporate these bottom-up processing (i.e. from sensory to more abstract representations) and top-down processing (i.e. from abstract to more sensory representations). In order to apply this idea to the current HTR model, composed of *feedforward reservoirs*, we could add *backwards reservoirs* (i.e. *feedback reservoirs*)[1]. These *backwards reservoirs* would be trained to predict a less abstract task (of the layer $n-1$) given a more abstract task (of the layer n). This would enable these *backwards reservoirs* to predict and update low-level representations based on more high-level representations. Most importantly, this could enable to not only *predict* but also to *postdict* [31] low-level outputs: this corresponds to predict the past given current information, i.e. update previous beliefs or perceptions. The integration of both *prediction* and *postdiction* in an architecture reminds the ability of bi-LSTMs to use both past and future input features [32]. Consequently, the output representations may need to be updated in order to use a kind of *a-temporal* representation of readouts: i.e. representing outputs for $t-n$, t and $t+n$ for any time step n, instead of just representing the current output at time step t.

The proposed HTR architecture is a promising first step towards general hierarchical modeling of language comprehension and production starting from speech signal. Further works in this line of research could focus on the addition of more abstract layers to perform tasks such as sentence chunking/segmentation, Name Entity Recognition, Sentiment Analysis or Semantic-Role Labelling [33]. Moreover, because the long-term goal of this architecture is to model brain processes, thus is not limited to speech or natural language processing, but sufficiently general to be applied to a variety of tasks, such as gesture recognition or sensorimotor learning.

REFERENCES

[1] M. H. Christiansen et al. *Creating language: Integrating evolution, acquisition, and processing*. MIT Press, 2016.

[2] I. Sutskever et al. Sequence to sequence learning with neural networks. In *NIPS*, pp. 3104–3112, 2014.

[3] D. Bahdanau et al. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] A. Vaswani et al. Attention is all you need. In *NIPS*, pp. 5998–6008, 2017.

[5] X. Hinaut et al. Exploring the acquisition and production of grammatical constructions through human-robot interaction with echo state networks. *Frontiers in Neurorobotics*, 8, 2014.

[6] J. Twiefel et al. Using Natural Language Feedback in a Neuro-inspired Integrated Multimodal Robotic Architecture. In *Proc. of RO-MAN*, New York City, USA, 2016.

[7] X. Hinaut and J. Twiefel. Teach your robot your language! trainable neural parser for modelling human sentence processing: Examples for 15 languages. *IEEE TCDS*, 2019.

[8] J. Hatori et al. Interactively picking real-world objects with unconstrained spoken language instructions. In *IEEE ICRA*, May 2018.

[9] F. Triefenbach et al. Acoustic modeling with hierarchical reservoirs. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(11):2439–2450, November 2013.

[10] A. Graves et al. Speech recognition with deep recurrent neural networks. In *IEEE ICASSP*, pp. 6645–6649, 2013.

[11] J. K. Chorowski et al. Attention-based models for speech recognition. In *NIPS*, pp. 577–585, 2015.

[12] N. T. Markov and H. Kennedy. The importance of being hierarchical. *Current Opinion in Neurobiology*, 23(2):187–194, April 2013.

[13] J. Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.

[14] S. E. Hihi and Y. Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *NIPS*, pp. 493–499, 1995.

[15] M. Hermans and B. Schrauwen. Training and analysing deep recurrent neural networks. In *NIPS*, pp. 190–198, 2013.

[16] C. Gallicchio et al. Deep reservoir computing: a critical experimental analysis. *Neurocomputing*, 268:87–99, 2017.

[17] C. Gallicchio et al. Design of deep echo state networks. *Neural Networks*, 108:33 – 47, 2018.

[18] D. Verstraeten et al. An experimental unification of reservoir computing methods. *Neural Networks*, 20(3):391–403, 2007.

[19] M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[20] F. Triefenbach et al. Phoneme recognition with large hierarchical reservoirs. In *NIPS*, pp. 2307–2315, 2010.

[21] J. Garofolo et al. Timit acoustic-phonetic continuous speech corpus. *Linguistic Data Consortium LDC93S1*, 1993.

[22] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical Report 148, German National Research Center for Information Technology GMD, Bonn, Germany, 2001.

[23] H. Jaeger et al. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3):335–352, 2007.

[24] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.

[25] C. Gallicchio. Chasing the echo state property. In *ESANN*, 2018.

[26] C. Gallicchio and A. Micheli. Echo state property of deep reservoir computing networks. *Cognitive Computation*, 9(3):337–350, May 2017.

[27] J. Twiefel et al. Syntactic reanalysis in language models for speech recognition. In *IEEE ICDL-EpiRob*, 2017.

[28] X. Hinaut and P. Dominey. Real-time parallel processing of grammatical structure in the fronto-striatal system: a recurrent network simulation study using reservoir computing. *PLoS ONE*, 8(2):e52946, 2013.

[29] X. Hinaut. Which input abstraction is better for a robot syntax acquisition model? phonemes, words or grammatical constructions? In *IEEE ICDL-EpiRob*, September 2018.

[30] X. Hinaut and M. Spranger. Learning to parse grounded language using reservoir computing. In *IEEE ICDL-EpiRob*, August 2019.

[31] A. Hanuschkin et al. A hebbian learning rule gives rise to mirror neurons and links them to control theoretic inverse models. *Frontiers in Neural Circuits*, 7, 2013.

[32] Z. Huang et al. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[33] J. Twiefel et al. Semantic role labelling for robot instructions using Echo State Networks. In *ESANN*, 2016.

[1]The word *feedback* could be misleading because already used in Reservoir Computing terminology. Thus, we replace the word *feedback* by *backwards* in the following discussion.