Hierarchical-Task Reservoir for Online Semantic Analysis from Continuous Speech

Luca Pedrelli, Xavier Hinaut

Hierarchical-Task Reservoir for Online Semantic Analysis from Continuous Speech

Luca Pedrelli

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

Xavier Hinaut

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

Abstract—In this paper, we propose a novel architecture called Hierarchical-Task Reservoir (HTR) suitable for real-time applications for which different levels of abstraction are available. We apply it to semantic role labeling based on continuous speech recognition. Taking inspiration from the brain, that demonstrates hierarchies of representations from perceptive to integrative areas, we consider a hierarchy of four sub-tasks with increasing levels of abstraction (phone, word, part-of-speech and semantic role tags). These tasks are progressively learned by the layers of the HTR architecture. Interestingly, quantitative and qualitative results show that the hierarchical-task approach provides an advantage to improve the prediction. In particular, the qualitative results show that a shallow or a hierarchical reservoir, considered as baselines, do not produce estimations as good as the HTR model would. Moreover, we show that it is possible to further improve the accuracy of the model by designing skip connections and by considering word embedding in the internal representations. Overall, the HTR outperformed the other state-of-the-art reservoir-based approaches and it resulted in extremely efficient w.r.t. typical RNNs in deep learning (e.g. LSTMs). The HTR architecture is proposed as a step toward the modeling of online and hierarchical processes at work in the brain during language comprehension.

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

I. INTRODUCTION

The number of models trained with End-to-End training over sequences has increased in the past years, in particular for speech recognition and natural language processing (NLP). However, it is usually costly in training time and data, while the training is performed offline. Conversely, human brains are able to process sentences online and learn incrementally to understand languages. Children start understanding and talking with much less data than such deep learning applications. One of the differences is that the human brain is likely to learn various levels of abstractions (e.g. phoneme, word, part-of-speech, semantic role of a group of words) in an incremental fashion instead of an end-to-end training. This hierarchical building of language building blocs is probably what enables children to learn quickly with little data.

Recently, deep learning networks have created a break-through in object and speech recognition. Latent represen-

tations of words and sentences, such as Word2Vec [1], and subsequent developments, such as transformers like BERT [2], enabled important progress on language modeling and natural language processing (NLP). However, no equivalent breakthrough happened towards the understanding of how the brain performs similar functions. This is probably due to the gap in the learning mechanisms between deep learning and brains.

Several mechanisms are subject to debates about biological plausibility. However, relying on back-propagation learning is often not considered plausible, especially if the gradient needs to go backwards through several layers. Back-propagation through time (BPTT) makes the implausibility a step further, as it needs to unfold time, which means to virtualise it as a spatial dimension in order to train e.g. a recurrent neural network (RNN).

Modeling brain processes, from raw acoustic signals up to language understanding, are a long-term research project. There is no such multi-level and hierarchical model on language today: this is an important shortage for the neuro-linguistic and psycho-linguistic communities. If one considers a model at one level only (e.g. sentence level), it means that this sentence model is agnostic to the processes going on until the word recognition. Thus, the modeler needs to make arbitrary assumptions (usually very simplified) on the dynamics of the hierarchical processes going on, in more perceptive layers, from raw speech processing to word recognition. Kröger et al. [3] did an interesting neurocomputational model of speech perception and production that spans on multiple levels, but only until phonemic map.

Christiansen & Chater propose that the brain is in the Now or Never Bottleneck [4] when processing a stimulus (e.g. an utterance): it is forced to extract the necessary information as soon as possible, otherwise, the information will be lost. Thus, the rich perceptual input needs to be recoded as it arrives, in order to capture the key elements of the sensory information [4]. These compressed (or "chunked") representations are abstractions of inputs (filtering out the details) rather than predictions encoding all the fluctuations of fast incoming inputs. Memory limitations also apply to these recoded representations; hence the brain needs to chunk the compressed representations into

multiple levels of representation of increasing abstraction in perception, and decreasing levels of abstraction in action [4]. Therefore, each sequence of chunks at one level will be encoded as a single chunk to a higher level. In summary, they suggest that the brain must implement this hierarchical "Chunk and Pass" [4] mechanism to solve the "Now or Never Bottleneck" problem.

State-of-the-art tools in NLP have little in common with the dynamical processes happening in our brains when reading a sentence. Most deep learning approaches for NLP use RNNs inside their architectures in order to model the temporal dependencies of the time-series input, and eventually, to implement a language model depending on the field of the task [5]–[12]. A way to address long-time dependencies between words is the use of bidirectional architectures and attention mechanisms [9], [10], [13]. In general, the whole sentence needs to be parsed before producing an output. Our brain processes a sentence in an online and *anytime* fashion: we are able to partially understand the sentence (and even predict it) before it ends.

For example, for speech applications in human-robot interaction [14]–[16], it is common to parse sentences based on a hand-written grammar parser, while the speech is processed with cloud speech APIs. Speech recognition modules based on sequence transduction approaches [7], [17], [18] have two main limitations: 1) the maximal length of the sentences need to be known beforehand and 2) the whole sentence needs to be parsed in order to produce the phone recognition of the input signal. Therefore, it puts time constraints for real-time human-robot interactions. Additionally, it is not suitable to model biologically plausible cognitive processes at work.

The brain is organised in hierarchical structures: for instance, a visual stimulus will go through the primary visual areas one after another: LGN, V1, V2, V3, V4, and so on. Already in the early 90's, Felleman & Van Essen [19] found a hypothetical global brain hierarchy. The deep learning approaches (DL), in particular the Convolutional Neural Networks (CNN), took successful inspiration from it. However, one could say that it is only a "shallow" inspiration because the brain is processing information in a much more dynamic way than how CNNs work. This is exemplified by the presence of *feedforward* and *feedback* connections in such brain hierarchies [20] (e.g. there are strong feedback connections from area V4 to area V2). A similar hierarchy starting from primary auditory areas has been proposed [21].

Although CNNs were successfully used to predict human brain fMRI responses [22], deep Recurrent Neural Networks (RNNs) [23]–[25] seem a better choice to really model hierarchical brain dynamics. They intrinsically develop hierarchical and distributed temporal features [26]–[28]. However, from the learning mechanisms point of view, the use of back-propagation in deep neural networks makes these approaches not biologically plausible. An interesting alternative to RNN back-propagation training is the Reservoir Computing (RC) paradigm [29], [30]. Recently, hierarchical [8], [17] and deep reservoir architectures [27], [28] achieved state-of-the-art re-

sults. In particular, Hierarchical Reservoir Computing (HRC) architectures obtained good results in speech recognition field [17].

Given such aspects, we propose a new architecture for semantic analysis from audio speech, called Hierarchical-Task Reservoir (HTR), with the following features: a) a model suitable for efficient real-time applications, b) a model able to learn progressively more abstract sub-tasks through a layers hierarchy, c) a model able to develop a hierarchical temporal representation and d) a model suitable for linguistic analysis in the neuroscience field.

First, we extend the speech recognition dataset TIMIT [31] building a novel corpus computing the Semantic Role Labeling (SRL) of the sentences pronounced in the audio speech. Then, we evaluate and compare HTR architectures on the SRL task. Finally, we quantitatively analyze the dynamics progressively developed in the HRT layers starting from the speech signal, provided as MFCC (Mel Frequency Cepstrum Coefficients). We expect the hierarchical-task architecture to enforce various abstraction levels of information through the different layers. One aim is to provide a richer representation of the input signal that could be denoised at different abstraction levels, instead of just denoising the raw signal. For this reason, HTR should achieve a better performance w.r.t. a 1-layered architecture. Since from POS tags there is not all the information to predict SRL tags, the prediction provided by the last layer (SRL from POS) could be difficult. However, the information sequentially carried by the layers could be enough to improve the prediction.

This work represents an extension of the preliminary studies regarding language comprehension through anytime POS tagging with a biologically plausible architecture [32]. In this paper, we introduce a novel real-time application based on SRL extending the Hierarchical-Task Reservoir architecture. Accordingly, we also extended the TIMIT [31] corpus by computing the SRL tags for all sentences. Moreover, we also experimentally studied the effect of word embedding and skip connections (in this case, between word representations and SRL task) on the prediction quality of the HTR architecture.

This paper is organized as in the following. In Section II, we describe the ESN and HRC architectures within RC. In Section III, we propose the HTR architecture. In Section IV, we define the method based on anytime semantic analysis from continuous speech. In Section V, we evaluate the HTR architectures on the anytime semantic analysis. Finally, we discuss the results of the analysis in Section VI.

II. RESERVOIR COMPUTING

Within Recurrent Neural Networks (RNNs), Reservoir Computing (RC) [33] represents a biologically plausible and efficient framework for architectural design. In particular, Echo State Networks (ESNs) are a class of RNNs implemented according to the RC framework in which the recurrent layer (i.e. the *reservoir*) is non-linear, randomly initialized and left untrained. Within ESN architectures, in this work, we consider the Leaky Integrator Echo State Network (LI-ESN) [34].

Figure 1 shows an example of LI-ESN architecture composed of a non-linear reservoir and a linear output (i.e. the *readout*). The following formula defines (omitting the bias for the ease
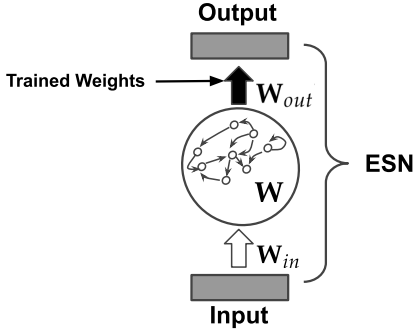

Fig. 1. The Echo State Network model.

of notation) the computation of the state at time step t:

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

where $\mathbf{x}(t) \in \mathbb{R}^{N_R}$ is the state vector at time step t, $\mathbf{u}(t) \in \mathbb{R}^{N_U}$ is the input vector at time t, $\mathbf{W} \in \mathbb{R}^{N_R \times N_R}$ is the matrix of the recurrent weights, $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_R \times N_U}$ is the matrix of the input weights, $a \in [0,1]$ is the leaking rate parameter and $\mathbf{tanh}$ is the activation function represented by the hyperbolic function. The recurrent weights in $\mathbf{W}$ are randomly initialized and rescaled according to the echo state property (ESP) [35]. Typically, in order to achieve the ESP, the maximum absolute eigenvalue of $\mathbf{W}$ (i.e. the spectral radius of $\mathbf{W}$) is rescaled to be less than 1. Moreover, in practical applications it is shown that the ESP can be obtained also with a spectral radius $\rho \geq 1$ [36]. The input weights in $\mathbf{W}_{\text{in}}$ are randomly initialized in order to obtain a specific euclidean norm σ of the matrix $\mathbf{W}_{\text{in}}$. The following formula computes the output of LI-ESN:

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

$\mathbf{W}_{\text{out}}$ is the matrix of the output weights and $\mathbf{y}(t) \in \mathbb{R}^{N_Y}$ is the output vector at time step t. The output layer is the only part of the ESN that is trained, in particular, the training consists in finding the free parameters in $\mathbf{W}_{\text{out}}$ through linear regression approaches. In the following, we use the term *ESN* to refer to the LI-ESN model.

Hierarchical RC (HRC) [17] is a class of RNNs characterized by a hierarchy of layers in which each layer is composed of an ESNs. In this architecture, each layer is trained starting from the output of the previous layer. In this way, each layer can correct the error resulting from the previous layer. In general, a hierarchical recurrent network is intrinsically (prior to learning) able to develop multiple time-scale dynamics as shown in Deep Reservoir Computing [27], [28].

III. HIERARCHICAL-TASK RESERVOIR

In this work, we introduce a novel hierarchical model characterized by a stack of layers called Hierarchical-Task Reservoir (HTR). Each layer is an ESN trained on a different task. The main idea is to consider progressively more abstract

tasks addressed by the layers. In this way, the deep recurrent architecture can impose (by training) a progressively more abstract (the labels are considered with a decreasing level of frequencies) representation of the input signal. The whole architecture is trained as a pipeline of tasks. The last layer addresses the final task that we aim to solve. The first ESN is trained on the first task by considering the input signal. Then each ESN is trained by considering as input the output of the previous ESN. The whole recurrent architecture is trained in pipeline fashion from the first to the last task. With respect to HRC presented in [17] there are two main differences: (i) HTR addresses a hierarchy of different tasks instead of addressing the same task in each layer as in HRC. Accordingly, the whole architecture can learn a progressively different representation of the input signal. (ii) HTR optimizes the hyperparameters of each layer instead to fix input scales and leak rates as in HRC. Accordingly, each task is optimized with different hyperparameters.

Figure 2 shown the main HTR architecture used in this work. The following formula computes the state of each layer $l = 1, ..., N_L$:

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

where $\mathbf{x}^{(l)}(t) \in \mathbb{R}^{N_R}$ is the state vector at time step t of layer l, $\mathbf{W}^{(l)} \in \mathbb{R}^{N_R \times N_R}$ is the matrix of the recurrent weights of layer l, $\mathbf{W}_{\text{in}}^{(l)} \in \mathbb{R}^{N_R \times N_U^{(l)}}$ is the matrix of the input weights of layer l, $a^{(l)} \in [0,1]$ is the leaking rate parameter and $\mathbf{f}$ is the activation function ($\mathbf{f}$ = tanh in this case). The input $\mathbf{i}^{(l)}$ of the layer l is computed as follows:

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

where $\mathbf{u}^{(1)} \in \mathbb{R}^{N_U^{(1)}}$ is the input vector of dimension $N_U^{(1)}$ and $\mathbf{y}^{(l-1)}(t) \in \mathbb{R}^{N_U^{(l)}}$ is the output vector of layer $l-1$ of dimension $\mathbb{R}^{N_U^{(1)}}$. The computation of the output of layer l is defined in the following equation:

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

where $\mathbf{W}_{\text{out}}^{(l)}$ is the matrix of the output weights. The recurrent weights in $\mathbf{W}^{(l)}$ are randomly initialized and rescaled to fix a spectral radius $\rho^{(l)}$. $\mathbf{W}_{\text{in}}^{(l)}$ is randomly initialized in order to obtain an euclidean norm $\sigma^{(l)}$. It is worth mentioning that, in deep recurrent architectures, the spectral radius $\rho^{(l)}$ and the input norm $\sigma^{(l)}$ are crucial hyperparameters to control the stability of the network [37]. Each layer is independently trained by finding the output weights $\mathbf{W}_{\text{out}}$ through ridge regression approaches as in standard RC.

The basic HTR model considered in our experiments is SP⇒PH⇒WD⇒POS⇒SRL. For each task (PH, WD, POS, SRL) the outputs are expressed as one-hot encoding if not stated otherwise. When considering word embedding we notify it with *WE* (instead of WD). (i) The model first estimates the phones from the input speech audio (the task **SP⇒PH**). (ii)

Then, **ESN 2** estimates words from the phones estimated by **ESN 1** (the task **PH⇒WD**). (iii) The **ESN 3** estimates the POS tagging from the words estimated by **ESN 2** (the task **WD⇒POS**). (iv) Finally, **ESN 4** estimates the SRL from the POS tagging estimated by **ESN 3** (the task **POS⇒SRL**). The procedure of HTR optimization is described in Algorithm 1, $N_{Configs}$ represents the number of hyperparameter configurations used in the random search. In the following, we consider $N_{Configs} = 100$ for all experiments.
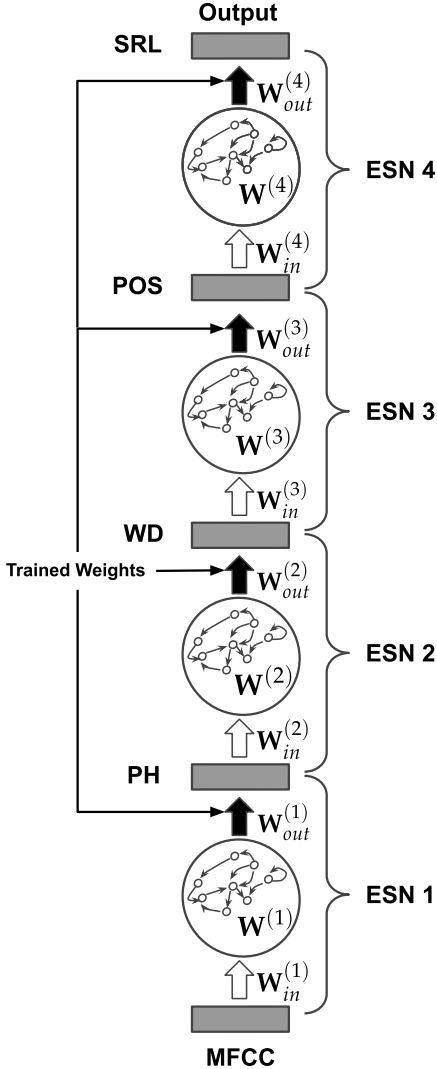
Task	Description
SP⇒PH	estimate phones from audio speech
PH⇒WD	estimate words from estimated phones
WD⇒POS	estimate POS from estimated words
POS⇒SRL	estimate SRL from estimated POS
SRL⇒SRL	estimate SRL from estimated SRL
SP⇒SRL	estimate SRL from estimated speech
PH⇒SRL	estimate SRL from estimated phones
WD⇒SRL	estimate SRL from estimated words
PH⇒WE	estimate word embeddings from estimated phones
WE⇒POS	estimate POS from estimated word embeddings
WE⇒SRL	estimate SRL from estimated word embeddings

Fig. 2. **The base HTR model considered in our experiments (SP⇒PH⇒WD⇒POS⇒SRL)**. First **ESN 1** estimates the phones from the input speech audio. Then, **ESN 2** estimates words from the phones estimated by **ESN 1**). The **ESN 3** estimates the POS tagging from the words estimated by **ESN 2**. Finally, **ESN 4** estimates the SRL from the POS tagging estimated by **ESN 3**. In some experiments, one-hot encoding of words (**WD**) are replaced by word embedding (**WE**).

IV. SEMANTIC ANALYSIS FROM AUDIO SPEECH

Here, we present a novel anytime approach based on Semantic Role Labeling (SRL) from audio speech. The aim of the task is to classify the semantic role of the pronounced word in the input audio in real-time each 10 ms.

In this work, we consider a hierarchical architecture that addresses a pipeline of tasks starting from speech audio. Since we need to start from a continuous speech recognition task, we consider the TIMIT [31] dataset. This corpus is composed of 630 American speakers. The training set is composed of 540 speakers and the test set is composed of 90 speakers. The validation set is composed of the last 135 speakers of the training set. Each speaker pronounces 10 sentences. The audio is labeled each 10 ms by 61 phone classes and by 6012 word classes. The number of considered phones are reduced from 61 to 51 as performed in [17]. In the case of word classes, we only kept the first 50 most frequent labels. The other words are grouped in a single label called out-of-vocabulary (OOV). Moreover, in order to improve the quality of the representation we also considered a word embedding trained on *Wikipedia* by using *fastText* [38] tools. To have a fair comparison with the one-hot-encoding approach used to address the 50 most frequent words, we produced a word embedding with dimension 50.

We extended the dataset by computing the POS tagging by using *SpaCy* [39] tools for each sentence with a total of 17 grammatical elements. The POS labeling 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 "h" is considered as an additional class in correspondence with silence frames in the speech signal.

Moreover, in order to form the SRL dataset, we used *AllenNLP* [40] tools computing the SRL of the TIMIT sentences with a total of 27 labels. Since in SRL there is a different labeling for each verb, in this work we consider a different classifier for each verb. Considering the order of the verbs in the sentence from the left to the right, we associate the i_{th} verb of the sentence to the i_{th} classifier (i.e. a readout in the case of ESN) with a maximum of 6 verbs considered. Accordingly, the i_{th} classifier is trained on the labels of the i_{th} verb. If the i_{th} verb is not present in the sentence, its labels are "X". Finally, as in the case of the other tasks, an additional label "h" represents 10ms of silence in the input audio. The input audio

is preprocessed through Mel Frequency Cepstral Coefficients (MFCC) as in [17] with a hamming window of 25 ms and a window shift of 10ms. For each 10ms, the Mel Frequency Cepstral Coefficients (MFCC) [17] algorithm computes the 39 components (including 13 $\Delta1$ and 13 $\Delta2$) used as input for the architectures by using a Fourier transform approach (see [17] for the full setup on MFCC preprocessing of TIMIT data). Thus, each input sequence is composed of a vector of 39 components for each frame of 10ms (e.g. an audio speech of 1 second is represented by an input sequence of 100 time steps and each time step is composed of a vector of 39 components).

Table I shows the list of the tasks addressed by model architectures in the next experiments.

As an evaluation metric for the models, we consider the frame error rate (FER) by computing the ratio of the not correctly classified frames on the total number of frames. Table VIII shows the hyperparameter ranges used in the random search for the model optimization performed for each HTR layer. For each configuration of hyperparameters, we randomly initialize 5 models (called *guesses*). Finally, the result achieved by a configuration is the average of the results achieved by the 5 guesses.

In order to design a real-time approach with a good trade-off between efficiency and accuracy and to perform several experiments on a medium/big dataset, we fix the number of recurrent units at $N_R = 1000$ for each layer.

Table II shows the architectures used for the experimental comparison. In particular, HTR is the proposed model introduced in Section III. While, HRC and ESN represent the state-of-the-art RC approaches described in Section II.

Model	Hierarchy
HTR (proposed)	**SP⇒PH ⇒WD⇒POS⇒SRL**
HRC	SP⇒SRL⇒SRL⇒SRL⇒SRL
ESN	SP⇒SRL

For the ESN model, we used 4000 units in order to have the same number of internal units as in the HRC and HTR architectures (which have 1000 units in each of the four layers).

V. RESULTS

A. Quantitative Comparison on Task SRL

In this section, we show the experimental comparison between HTR (Hierarchical-Task Reservoir, the proposed architecture), HRC (hierarchical reservoir baseline), ESN (shallow reservoir baseline) and Long Short Term Memory (a typical literature RNN approach called LSTM) on the SRL task. The test results achieved by HTR, HRC, ESN and LSTM are shown in Table III. In particular, HTR outperforms the other RC approaches achieving a test error of 22.32% while HRC achieved a test error of 23.61% and ESN achieved a test error 23.44%. Interestingly, HRC is not able to improve the

TABLE III
THE ERRORS ACHIEVED IN TEST SET BY HTR, HRC, ESN AND LSTM ON TASK SRL.

Model	Test FER
HTR	**22.32(0.16)%**
HRC	23.61(0.14)%
ESN	23.44(0.02)%
LSTM	33.18(0.23)%

result achieved by ESN. This highlights that a pipeline of ESN modules that address the same SRL task is not able to improve the results obtained by the ESN with a single layer composed of the same number of total recurrent units. Conversely, the learning based on hierarchical-task, addressing progressively more abstract tasks from the first to the last layer, allows the model to significantly improve the test error on task SRL obtaining 1.29 points more than HRC and 1.12 points more than ESN.

In order to have a comparison with a typical fully-trained RNN approach used in deep learning, we considered an LSTM recurrent network which is used in literature [8] as a baseline comparison with hierarchical RC architectures on speech recognition tasks. As described in [41] it is difficult to define a fair comparison between randomized and fully-trained RNNs. However, the setting proposed in [41] is adapted to show the potential of RC-based approaches w.r.t. typical literature approaches. Given such aspects, we performed an experimental assessment considering the same setting used in [41] for the optimization of the LSTM model. Table III shows the test FER obtained by LSTM on the SRL task. The LSTM model (with the same number of free parameters w.t.r. the RC approach) obtained a test FER of 33.18%. This result highlights the effectiveness of the RC approaches in terms of performance that achieved significantly lower Test FERs (see Table III) than the LSTM approach. Moreover the LSTM model spent 4813.83 seconds for the training process w.r.t. the HTR (our approach) model which only spent 217.08 seconds. We can conclude that the proposed approach is more efficient than typical fully-trained RNNs in the literature.

B. Quantitative Results of HRC by Increasing Layers

Here, we present the test error achieved by HRC on the SRL task by increasing the layers number. As we can see from Table IV the HRC with 1, 2, 3 and 4 number of layers achieved a test error of 23.85%, 23.58%, 23.50% and 23.61% on the SRL task. As we can note, the test error obtained by HRC

TABLE IV
THE ERRORS ACHIEVED IN TEST SET BY THE HRC MODEL INCREASING THE LAYERS NUMBER ON TASK SRL.

Hierarchical Reservoir Computing	Test FER
SP⇒SRL	23.85(0.11)%
SP⇒SRL⇒SRL	23.58(0.18)%
SP⇒SRL⇒SRL⇒SRL	**23.50**(0.02)%
SP⇒SRL⇒SRL⇒SRL⇒SRL	23.61(0.14)%

improves until layer 3. After that, the error no longer improves

with 4 layers. This highlights that it is difficult to improve the Anytime SRL task just by considering a pipeline of ESNs without the additional abstraction of the tasks addressed in the layers of HTR.

C. Improve Results of HTR by Increasing Layer Abstraction

Here, we present the test error achieved by HTR on the SRL task by increasing the layers number. From Table IV we can see that the HTR with 1, 2, 3 and 4 number of layers achieved a test error of 23.85%, 23.06%, 22.34% and 22.32% on the SRL task. Interestingly, these results highlight that HTR is able to have a significant progressive improvement with the increasing of layers. From Figure 7, we can see

TABLE V
THE ERRORS ACHIEVED IN TEST SET BY HTR MODEL
INCREASING THE LAYERS NUMBER ON TASK SRL.

Hierarchical-Task Reservoir	Test FER
SP⇒SRL	23.85(0.11)%
SP⇒PH⇒SRL	23.06(0.11)%
SP⇒PH⇒WD⇒SRL	22.34(0.33)%
SP⇒PH⇒WD⇒POS⇒SRL	**22.32**(0.16)%

the comparison between the test errors achieved by HTR and HRC with a number of layers that goes from 1 to 4. Note that, while the test error achieved by HRC quickly saturates, the test error achieved significantly decreases with increasing layers. In conclusion, these results show the crucial role of the hierarchical-task method, which addresses progressively abstract tasks, in improving the performance achieved on the final SRL task. However, from Figure 7 it is worth noting that the decrease of error obtained by HTR in the last layer (i.e. the POS⇒SRL task) is very small. This can be motivated by the fact that the information provided by the POS task alone is not sufficient to solve the SRL task. A possible solution is considering a skip connection between the WD layer to the SRL layer in order to add useful information to the last layer and improve the performance. A skip connection is a direct link between two layers that are not linked in the default hierarchical architecture.

D. Comparison of Prediction Quality by Increasing Layers

Here, we compare the output predictions performed by HTR and HRC on task SRL at the increasing of layers. In this example, we take into account the sentence "don't ask me to carry an oily rag like that". Concerning the SRL task we consider the labels relative to the verb "carry": "O O ARG0 O V ARG1 ARG1 ARG1 ARG1 ARG1" where *V* is the verb and *ARG0* and *ARG1* are the arguments of the verb where *ARG0* is the *Proto-Agent* and *ARG1* is the *Proto-Patient*. The Figure 3a shows the input signal represented by the MFCC of the audio speech. The Figures 3b, c, d and e represent the values of the HTR output by considering 1, 2, 3 and 4 number of layers, respectively.

The lines represented in Figures 3b, c, d and e are the components of the output of the HTR model with a different
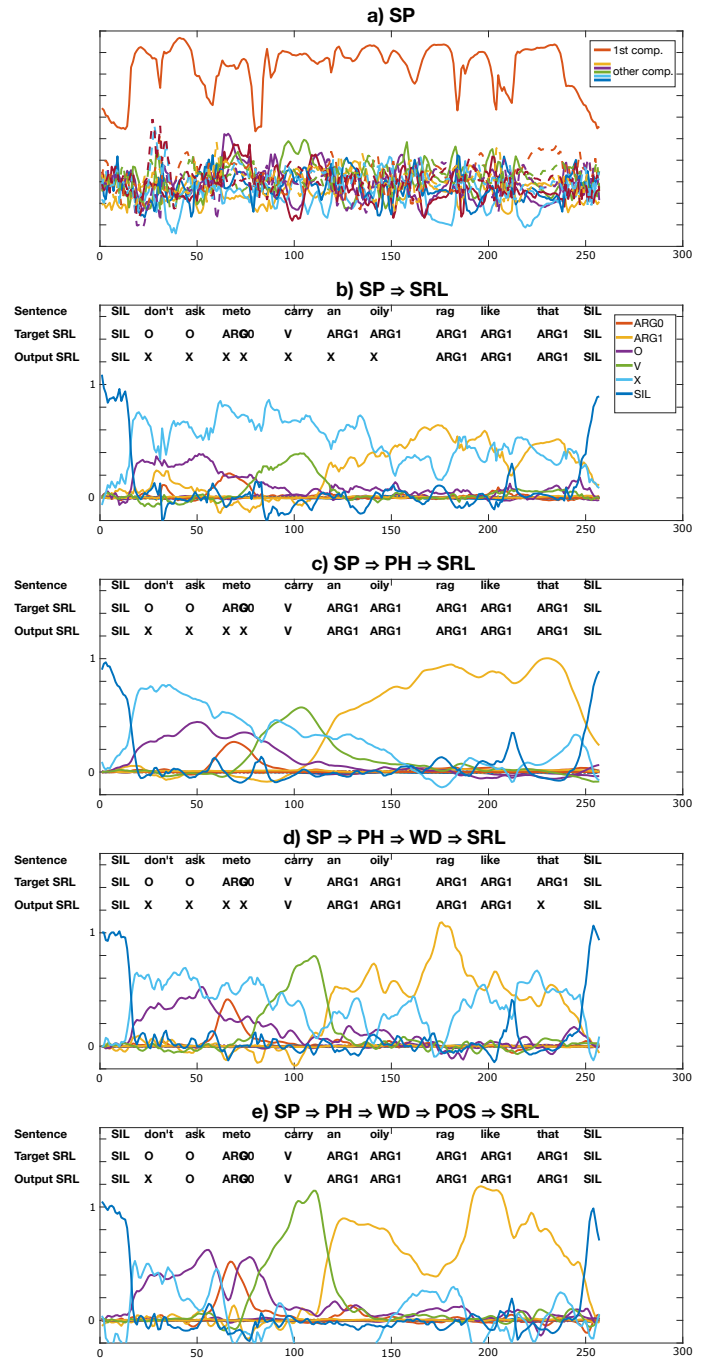

Fig. 3. **An example of the prediction components provided by the HTR model with a number of layers that goes from 1 to 4.** Figure a) represents the audio speech in the form of MFCC components. Figure b) represents the output predictions of the architecture SP⇒SRL (1 layer). Figure c) represents the output predictions of the architecture SP⇒PH⇒SRL (2 layers). Figure d) represents the output predictions of the architecture SP⇒PH⇒WD⇒SRL (3 layers). Finally, Figure e) represents the output predictions of the architecture SP⇒PH⇒WD⇒POS⇒SRL (4 layers). Time-steps are represented in the x-axis while the output values of the neural components are represented in the y-axis. Each component represents a class and for each time-step the model predicts the class related to the component with the maximum value.

number of layers. For each time-step, the output SRL estimated by the model is the class relative to the component with

the maximum value. The labels (i.e. the target ground truth) and the estimated classes over time are represented above the components values. Note that the estimations represented in Figure 3b suffer from uncertainty. Indeed, many prediction ranges are too flat to have a clear estimation. However, considering the estimations produced by the 2-layered HTR architecture, we note that the quality of the predictions is significantly improved due to a better separation of the prediction ranges. Finally, the estimation quality further improved by adding more layers as shown in Figure 3d and e. This highlights that the estimation quality is progressively improved considering more sub-tasks inside the HTR architecture before solving the final SRL task.

Concerning the HRC model, the Figures 8b, c, d and e represent the values of the HRC output on the SRL task by considering 1, 2, 3 and 4 number of layers, respectively. In this case, there is no significant improvement of predictions if we add layers in the HRC architecture.

Overall, the predictions performed by the HTR model (Figure 3e) have a significantly better quality compared with the predictions performed by the HRC model (Figure 8e). This confirms the quantitative experiments shown in Section V highlighting the importance to have a progression of different tasks in order to progressively improve the prediction. Note that Figures 3a, b and Figures 8a, b display exactly the same results, because the same reservoirs with the same hyperparameters were used.

E. Analysis of Layer Representations in HTR

Here, we qualitatively analyze the internal representations (i.e. the output of each layer) learned by HTR with 4 layers. Figures 4b, c, d and e represent the estimation values performed by the architecture SP⇒PH⇒WD⇒POS⇒SRL. Interestingly, the lower recurrent layers have a dynamic that is faster than the higher layers. This is expected since the frequency of the labels used to learn the sub-tasks is progressively lower in the higher layers. In other words, the learning based on a progressively lower frequency of labels forces a progressively lower frequency in the signals of the output layers.

Concerning the ability of HTR to correct errors between two different tasks, an interesting example can be seen in correspondence of the word label "carry" in the outputs of the HTR layers (see Figures 4c, d and e). Despite the prediction relative to the word is wrong (i.e. "OOV" instead of "carry") in layers 3 and 4 the model is able to correct the predictions performing the correct estimation for POS and SRL tasks ("VERB" and "V" labels). These results showed that the HTR model is able to correct the estimation in the higher layers by progressively extrapolating the information from the previous layers though the sub-task predictions are incorrect.

F. Skip Connections in HTR

Here, we consider the use of skip connections in the HTR architecture. In particular, from Figure 7 we can see that the improvement achieved by layer 4 is very small. The POS
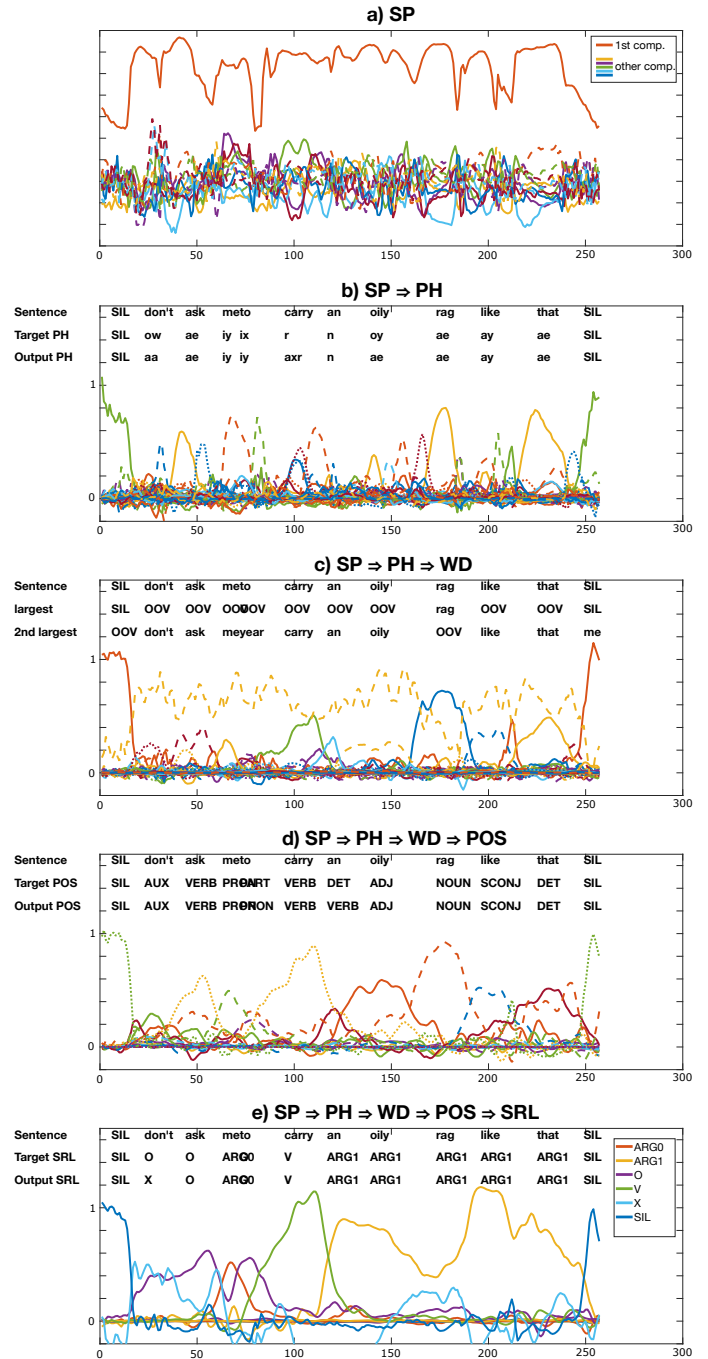

Fig. 4. **An example of the prediction components by each layer of the HTR model SP⇒PH⇒WD⇒POS⇒SRL (4 layers).** Figure a) represents the audio speech in the form of MFCC components. Figure b) represents the output predictions of the architecture SP⇒PH. Figure c) represents the output predictions of the architecture SP⇒PH⇒WD. Figure d) represents the output predictions of the architecture SP⇒PH⇒WD⇒POS. Finally, Figure e) represents the output predictions of the architecture SP⇒PH⇒WD⇒POS⇒SRL. Time-steps are represented in the x-axis while the output values of the neural components are represented in the y-axis. Each component represents a class and for each time-step the model predicts the class related to the component with the maximum value.

information alone could be not enough to improve the error on the SRL task. Then, we add skip connections from the

output of ESN 2 to the input of ESN 4 in order to carry the WD information to the POS⇒SRL layer. In this way, we can study if adding sub-tasks information can help the model to improve the prediction. In the implementation of this solution, the input of layer 4 is concatenated with the output of layer 2 in order to obtain the skip connection.

TABLE VI
THE ERRORS OBTAINED IN TEST SET BY HTR WITH SKIP CONNECTION ON TASK SRL.

Models	Test FER
SP⇒PH⇒WD⇒SRL	22.34(0.33)%
SP⇒PH⇒WD⇒POS⇒SRL	22.32(0.16)%
SP⇒PH⇒WDskip⇒POS⇒SRL	**21.93(0.03)**%

Table VI presents the results achieved by the HTR architecture with the use of the skip connection. Interestingly, the use of skip connection obtains an improvement of 0.39 FER points. It is worth mentioning that both POS information and skip connection are crucial to improve the performance since the architecture without the POS task (SP⇒PH⇒WD⇒SRL) obtains a worse result.

G. Word Embedding in HTR

Here, we experimentally study the effect obtained by the use of word embeddings (WE) in the HTR architecture instead of using one-hot-encoding (WD). Moreover, in order to exploit the WE information in the SRL task, as in the previous section, we consider skip connections from the output of ESN 2 to the input of ESN 4.

TABLE VII
THE ERRORS OBTAINED IN TEST SET BY HTR WITH WORD EMBEDDING AND SKIP CONNECTION ON TASK SRL.

Models	Test FER
SP⇒PH⇒WE⇒SRL	22.18(0.16)%
SP⇒PH⇒WE⇒POS⇒SRL	22.43(0.16)%
SP⇒PH⇒WEskip⇒POS⇒SRL	**21.50(0.03)**%

Table VII presents the results achieved by the model HTR with the use of WE and skip connections. In this case, the use of skip connections enables to obtain an improvement of 0.93 FER points. Overall, the use of WE and skip connections allows us to achieve an improvement of 0.43 points w.r.t. the use of WD and skip connection. Table IX (in Supp. Mat.) shows the hyper-parameters selected in the random search for each layer in the best HTR model SP⇒PH⇒WEskip⇒POS⇒SRL.

H. Qualitative analysis of Word Embedding in HRT

This section is described in Supplementary Material in subsection VII-A.

I. Qualitative analysis of Word Embedding and skip connections in HRT

This section is described in Supplementary Material in subsection VII-B.

VI. DISCUSSION

Inspiring from brain hierarchical organisation, both in terms of spatial organisation (anatomy) and processes abstraction, we propose a new reservoir architecture: the Hierarchical-Task Reservoir (HTR). For this purpose, we introduced a novel task for anytime semantic analysis from continuous speech recognition extending the TIMIT corpus. The aim of this task is to perform real-time classification (frame by frame) of the semantic roles of the utterance in the input audio stream. The HTR is composed of a hierarchy of layers in which each layer is composed of an ESN that addresses a different task (i.e. phone, word, POS and SRL) starting from the input of the previous layer.

The qualitative experiments highlight that sub-tasks learned to produce a progressively decreasing temporal frequency of labels when going towards the top of the hierarchy. This feature can help the HTR to progressively develop a meaningful level of abstraction of the input signal in order to improve the performance of the whole task.

Another advantage that we exploited in HTR is the possibility to directly combine a different kind of information from different tasks by using skip connections between layers. In particular, since we obtained a small improvement by adding the POS layer in HTR, we considered also an architectural variant with skip connection between the word layer and the SRL layer in order to avoid a possible bottleneck due to the POS layer. In this way, the SRL can be predicted starting from the word in addition to the POS tag. The results show that the use of skip connection significantly improves the performance and outperforms also the HTR with 3 layers (without POS layer). This highlights that the task hierarchy and the skip connection are both important to develop and provide a meaningful representation to improve the whole semantic analysis task.

Moreover, we additionally considered the embedding word representation in HTR in replacement of the use of one-hot-encoding with OOV word. The qualitative analysis of the prediction highlighted that the embedding layer allows the architecture to improve the quality of the SRL prediction. In particular, the quality of prediction is significantly enhanced by the use of skip connection between the output of ESN 2 (with the word embedding) and the input of ESN 4 (with the POS tagging).

The best HTR architecture (which combines both word embedding and skip connection) obtained 21.50% error. This is an interesting result considering the difficulty of the task that we proposed (starting from MFCC inputs), the difficulty of the TIMIT corpus (which contains a lot of variability due to numerous speakers that provide only 10 sentences each, and which contains complex sentences), and the imposed online answers the architecture have to provide (usually SRL is performed offline: the whole sentence is parsed before answering). This latter point is particular because at a given point in time (e.g. before the first verb is seen) the architecture often does not have enough information to give the correct

answer (e.g. the first noun-phrase of the sentence could be the agent (e.g. ARG0) or the recipient (e.g. ARG2) of the sentence). Thus, this could be easily changed if one looked for the best performance by permitting the architecture to wait until the end of the sentence before answering: the semantic roles classification would be enhanced by a good margin.

In order to experimentally assess the potential of the proposed RC-based approach, we performed an experimental comparison with the LSTM model which represents a typical RNN from the deep learning domain for continuous speech recognition suitable for online computation. Overall, the experimental results show that the proposed approach is significantly performant and extremely efficient w.r.t. the LSTM-based approach.

In general, the approaches introduced in this paper regarding HTR are interesting also from the point of view of typical deep learning models based on back-propagation. Indeed, the learning of different kind of abstractions in the layers is a key point in deep learning [24], [42]. Therefore, in future works the hierarchical-task approach could be combined with back-propagation, for instance by forcing different loss constraints in different layers on the basis of a hierarchy of tasks [26]. Hopefully, this approach could enable back-propagation to use less training data for comparable performance. Because, with the same number of samples (i.e. sentences), the hierarchical-task design provides more information to the internal representations. Moreover, the use of skip connections in deep RNN can also help the back-propagation process allowing the model to improve the performance [43], [44]. Therefore, the HTR can be considered as a sort of baseline for the design and the development of deep architectures based on a hierarchy of tasks and skip connections.

One can observe that the HTR outputs dynamics decrease in frequency across the layers, future work could try to make a hierarchy of tasks without providing the intermediate labels. Just by imposing a decreasing frequency constrain (i.e. a slower speed) on the output representations across layers without using intermediate labels. This idea is similar to previous work related to RNNs [26], [45].

Overall, the HTR model and the anytime semantic analysis task, introduced in this paper, are interesting tools for further studies regarding language comprehension in neuroscience approaches [46], [47] or for the implementation of a real-time human-robot interaction (HRI) [14]–[16], [48]. In future work, we will use this efficient HTR architecture in Human-Robot Interaction (HRI) context. Indeed, the availability of different levels of abstraction would provide more robust speech processing for real-time and noisy HRI environments.

Regarding the neuroscience field, in future work, it would be interesting to get more inspiration from neurobiological findings on brain hierarchy: in primate brains there are *feedforward* and *feedback* connections between brain areas of different abstraction levels [20]. Indeed, the 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 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* [49] 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 [50]. 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 a 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 or Sentiment Analysis. 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.

In a nutshell, we proposed a new way of training the hierarchical reservoir architecture which is more effective compared to the original training for our application: both quantitatively and qualitatively (given the increased robustness of the outputs obtained). In general, we could even argue that our approach provides quantitative and qualitative advantages by exploiting the hierarchical reservoir topology for applications that have an intrinsic hierarchical latent structure. The qualitative advantage is also provided by the interpretability of the results (e. g. the output dynamics) in the intermediate layers. Moreover, as we have shown, this HTR model can address the application we used, but it is not limited to such applications. Indeed, many applications that have latent representations that are hierarchical/compositional can be represented with hierarchical labels. In general, any application where one has to recognize something which consists of smaller parts (the edges and features of an object for vision for instance, and the same for audio). This is true for any kind of "gesture" recognition; gesture seen in a large sketch, for motor actions patterns or audio patterns, where parts of a continuous sequence can be chunked together to form a "gesture". In many applications these gestures are compositional, thus producing a hierarchy

---

[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.

of chunks/gestures that could be represented at different levels of abstraction. An obvious continuation of our approach to future work is the application to musical data, where audio inputs can be represented as note pitch, rhythm, chords, harmony, etc. Even more generally, the HTR approach can be used to modelize continuous and online applications based on sequences by changing the frequencies of the labels in each layer of the architecture. Similarly to the RC-HMM [8] triphone approach for speech recognition (i.e. three HMM states), we can consider a different label for each state of the Markov chain. Therefore in the HTR approach, the higher layers can be used to address fewer states in order to impose a decrease in the frequency of the labels in the higher layers. In this way, the HTR approach can provide the advantages shown in this work on a general time-series application.

ACKNOWLEDGMENT

REFERENCES

[1] T. Mikolov et al. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[2] J. Devlin et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] B. J. Kröger et al. Towards a neurocomputational model of speech production and perception. *Speech Communication*, 51(9):793–809, 2009.

[4] M. H. Christiansen and N. Chater. *Creating language: Integrating evolution, acquisition, and processing*. MIT Press, 2016.

[5] T. Mikolov et al. Extensions of recurrent neural network language model. In *ICASSP 2011*, pp. 5528–5531. IEEE, 2011.

[6] C. Chelba et al. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005, 2013.

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

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

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

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

[11] A. Kumar et al. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285, 2015.

[12] M. Luong et al. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015.

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

[14] 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.

[15] 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.

[16] 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.

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

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

[19] D. J. Felleman and D. C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. In *Cereb cortex*. Citeseer, 1991.

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

[21] J. H. Kaas and T. A. Hackett. Subdivisions of auditory cortex and processing streams in primates. *Proceedings of the National Academy of Sciences*, 97(22):11793–11799, 2000.

[22] A. J. Kell et al. A task-optimized neural network replicates human auditory behavior, predicts brain responses, and reveals a cortical processing hierarchy. *Neuron*, 98(3):630–644, 2018.

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

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

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

[26] Y. Yamashita and J. Tani. Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment. *PLoS Comput Biol*, 4(11):e1000220, 2008.

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

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

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

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

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

[32] L. Pedrelli and X. Hinaut. Hierarchical-Task Reservoir for Anytime POS Tagging from Continuous Speech. In *IJCNN 2020*.

[33] 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.

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

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

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

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

[38] P. Bojanowski et al. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[39] M. Honnibal et al. spaCy: Industrial-strength Natural Language Processing in Python, 2020.

[40] M. Gardner et al. AllenNLP: A Deep Semantic Natural Language Processing Platform. *CoRR*, abs/1803.07640, 2018.

[41] C. Gallicchio et al. Comparison between DeepESNs and gated RNNs on multivariate time-series prediction. In *ESANN 2019*, pp. 619–624. ESANN (i6doc. com), 2019.

[42] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pp. 818–833. Springer, 2014.

[43] R. K. Srivastava et al. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[44] K. He et al. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[45] S. El Hihi and Y. Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in neural information processing systems*, pp. 493–499, 1996.

[46] 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.

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

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

[49] 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.

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

VII. Supplementary Material

A. Qualitative analysis of Word Embedding in HRT

In this section, we evaluate the quality of the prediction by using the word embedding layer in HTR. Figure VII-A shows the output prediction provided by the architecture PS⇒PH⇒WE⇒POS⇒SRL. In this case (with the WE layer),
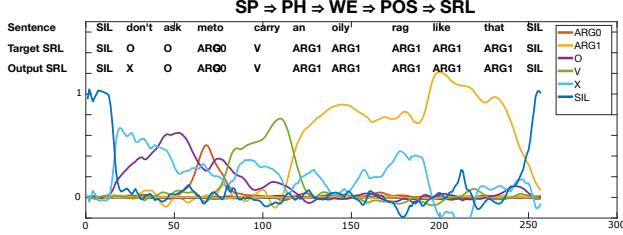

Fig. 5. **Outputs of HTR with the use of word embedding.** The Figure shows the output values of the architecture PS⇒PH⇒WE⇒POS⇒SRL.

there is a slight improvement of the quality of prediction w.r.t. to the use of the one-hot-encoding word layer (WD layer). In particular, the prediction are more regular and smooth w.r.t. the predictions of PS⇒PH⇒WD⇒POS⇒SRL (see Figures VII-A and 4e). Moreover, the HTR with word embedding presents a better certainty in some areas of the prediction (for instance in the right part of Figure VII-A).

B. Qualitative analysis of Word Embedding and skip connections in HRT

In this section, we evaluate the quality of the prediction combining word embedding and skip connection in HTR. Figure 6 shows the output prediction provided by the architecture PS⇒PH⇒WEskip⇒POS⇒SRL. Interestingly, we can
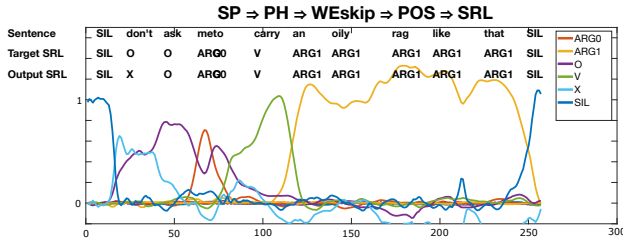

Fig. 6. **Outputs of HTR with the use of word embedding and skip connection.** The Figure shows the output values of the architecture PS⇒PH⇒WEskip⇒POS⇒SRL.

see that with the use of skip connection the quality of prediction is further improved. Indeed, the prediction components are more separated meaning an improvement of certainly. For instance, after some initial steps the predicted component is very high while the other components are very near zero (see Figure 6).

Table IX shows the hyper-parameters selected in the random search for each layer in the best HTR model SP⇒PH⇒WEskip⇒POS⇒SRL.

Algorithm 1 HTR Optimization

1: **procedure** OPTIMIZEHTR($\mathbf{i}^{(1)}$, $N_{Configs}$)
2:     **for** l in 1, ..., N_L **do**
3:         $\theta^{(l)} = $ initConfigs($N_{Configs}$)
4:                 ▷ initialize $N_{Configs}$
5:         HTR$^{(l)} = $ bestModel($\theta^{(l)}, l$)
6:                ▷ model selection on task l
7:         $\mathbf{i}^{(l+1)} = $ output(HTR$^{(l)}, \mathbf{i}^{(l)}$)
8:            ▷ output of layer l as input of layer $l+1$
    return HTR
9:               ▷ return the optimized HTR model

Hyper-parameter ranges	
spectral radius $\rho^{(l)}$	logarithmic distribution in [0.1, 10]
input norm $\sigma^{(l)}$	logarithmic distribution in [0.1, 10]
leaky integrator $a^{(l)}$	uniform distribution in [0.1, 1]
ridge (regularization) $\lambda^{(l)}$	sampling in $[10^0, 10^{-1}, ..., 10^{-8}]$

TABLE VIII
RANGE OF HYPER-PARAMETERS VALUES USED IN THE RANDOM SEARCH FOR THE MODEL SELECTION OF EACH HTR LAYER l.
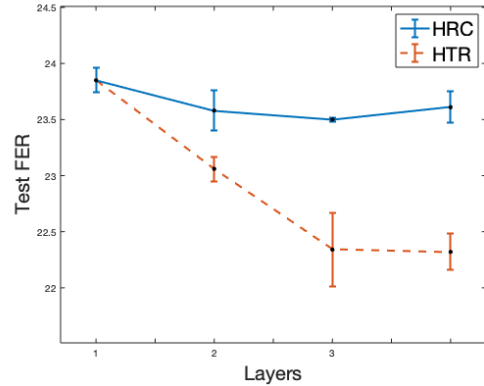

Fig. 7. **The errors achieved in Test set by HTR and HRC model increasing the layers number.** The standard deviations are represented by the vertical ranges.

TABLE IX
THE HYPER-PARAMETERS SELECTED FOR EACH LAYER IN THE BEST HTR MODEL SP⇒PH⇒WESKIP⇒POS⇒SRL.

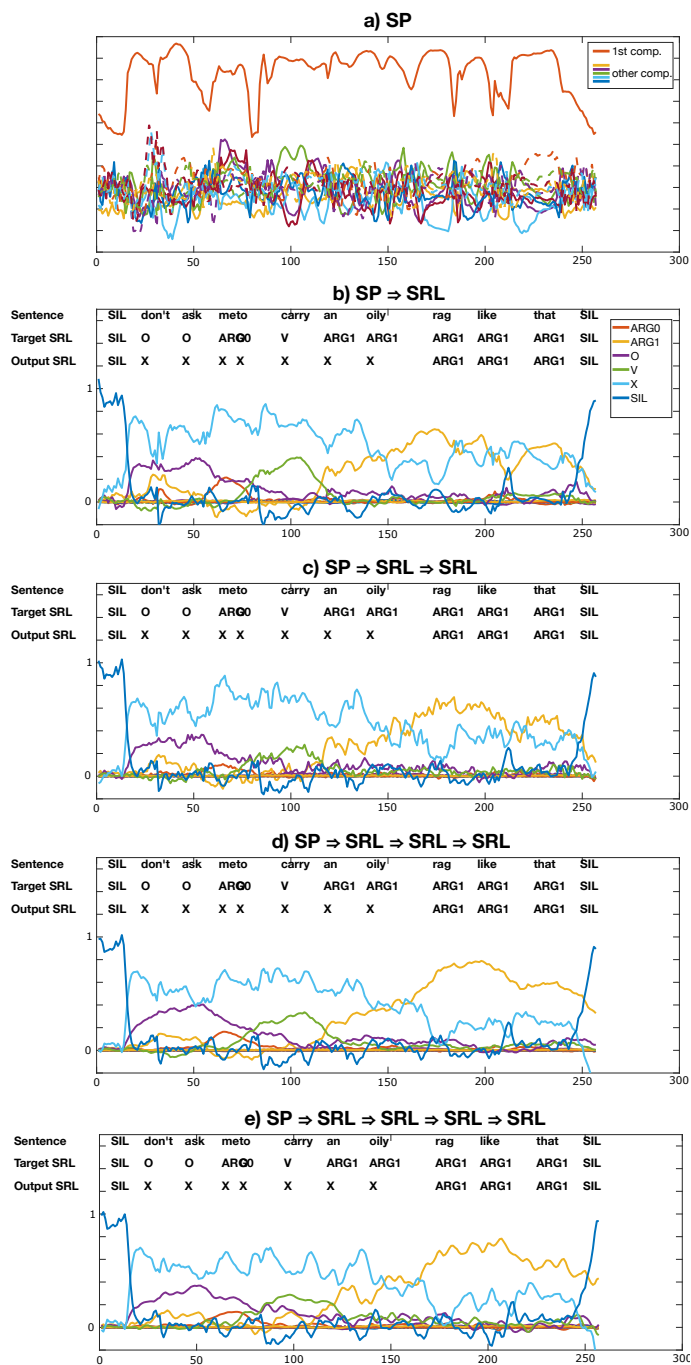| H.p.	$l=1$	$l=2$	$l=3$	$l=4$
$\rho^{(l)}$	0.8616	1.1786	1.2298	1.4277
$\sigma^{(l)}$	0.7210	8.8463	2.6722	8.0801
$a^{(l)}$	0.2961	0.4663	0.6632	0.2265
$\lambda^{(l)}$	$6.56 \cdot 10^{-3}$	$4.59 \cdot 10^{-5}$	$1.49 \cdot 10^{-5}$	$8.23 \cdot 10^{-7}$

Fig. 8. **An example of the prediction components provided by the HRC model with a number of layers that goes from 1 to 4.** Figure a) represents the audio speech in the form of MFCC components. Figure b) represents the output predictions of the architecture SP⇒SRL (1 layer). Figure c) represents the output predictions of the architecture SP⇒SRL⇒SRL (2 layers). Figure d) represents the output predictions of the architecture SP⇒SRL⇒SRL⇒SRL (3 layers). Finally, Figure e) represents the output predictions of the architecture SP⇒SRL⇒SRL⇒SRL⇒SRL (4 layers). Time-steps are represented in the x-axis while the output values of the neural components are represented in the y-axis. Each component represents a class and for each time-step the model predicts the class related to the component with the maximum value.