

Contextual Processing of Structured Data by Recursive Cascade Correlation

Alessio Micheli, *Member, IEEE*, Diego Sona, Alessandro Sperduti, *Member, IEEE*,

Abstract— We propose a first approach to deal with contextual information in structured domains by Recursive Neural Networks. The proposed model, i.e. Contextual Recursive Cascade Correlation (CRCC), a generalization of the Recursive Cascade Correlation (RCC) model, is able to partially remove the causality assumption by exploiting contextual information stored in frozen units. We formally characterize the properties of CRCC showing that it is able to compute contextual transductions and also some causal supersource transductions that RCC cannot compute. Experimental results on controlled sequences and on a real-world task involving chemical structures confirm the computational limitations of RCC, while assessing the efficiency and efficacy of CRCC in dealing both with pure causal and contextual prediction tasks. Moreover, results obtained for the real-world task show the superiority of the proposed approach versus RCC when exploring a task for which it is not known whether the structural causality assumption holds.

Index Terms— Contextual mapping, Cascade-Correlation, recurrent and recursive neural networks, neural networks for structured data, computational power, learning in structured domains.

I. INTRODUCTION

RECURSIVE Neural Networks, which generalize Recurrent Neural Networks to the processing of directed acyclic graphs, have been recently defined [1], [2], [3], [4]. These models are able to learn a mapping from a domain of positional directed acyclic graphs (note that sequences are included in this class), with labels attached to each node, to the set of real numbers. The basic idea behind the models is the extension of the concept of *unfolding* from the domain of sequences to the domain of directed ordered acyclic graphs.

Recursive Neural Networks, as well as almost all the Recurrent Neural Network models proposed in literature are based on the causality assumption. When considering sequences, the causality assumption states that the output of the network at time t_0 only depends on input at times $t \leq t_0$. In the framework of structure processing, a model is *causal* (i.e., it strictly satisfies the causality assumption) if the output for a given vertex of a directed acyclic graph only depends on the information conveyed by the current vertex and the vertexes descending from it. This assumption allows to use internal states of the network to memorize information about substructures.

Manuscript received ??, ??; revised ??, ?. This work has been partially supported by MIUR grants 2002093941_004 and 2001034475_006.

A. Micheli is with Dip. di Informatica, Università di Pisa, *micheli@di.unipi.it*, D. Sona is with SRA division, ITC-IRST, *sona@itc.it*, A. Sperduti is with Dip. di Matematica Pura ed Applicata, Università di Padova, *sperduti@math.unipd.it*.

Nevertheless, several prediction tasks involving items both in sequences and structured data domains, where we assume the availability of the whole sequence/structure at the time of processing, require processing of information from both the “past” and the “future”, i.e., contextual information. The DNA and proteins analysis, as well as language understanding, are examples of these tasks. Standard solutions to this problem, in the framework of sequence processing, involve feed-forward neural networks that look at the input through a fixed window of predefined size [5] [6]. These approaches, however, are not practical if a priori knowledge is not available on the “optimal” size of the window. Some authors suggested to solve the fixed size window problem, still in the case of sequences, by specific models which compute the output by combining information propagated from both the “past” and the “future”. This is performed spanning the sequence in the two directions. For example, in the recurrent model proposed in [7], the internal state is factorized into a forward state and a backward state. In particular the devised *Bidirectional Recurrent Neural Network* (BRNN) is composed of three sub-networks: one for computing the “past” information, one for computing the “future” information, and finally one sub-network which combines all the information to produce the output. A related approach has been proposed in [8].

A different approach has been introduced in [9] where the proposed model is a variant of the basic Recurrent Cascade Correlation [10], referred to as *Bi-causal Recurrent Cascade Correlation* (BRCC). Actually, when training a Recursive Cascade Correlation Network, hidden units are frozen one by one as new units are added. Since weights of frozen units are not allowed to change, it is possible to use the state information of the frozen units to also analyze an internal representation of the “future” inputs. When training a new hidden unit the information stored in frozen units can be accessed. In this way, when processing a sequence s at a time t , it is possible to use the stored activations for all the following subsequences of s ,

$$s_{[0,1]}, s_{[0,2]}, \dots, s_{[0,t-1]}, s_{[0,t]}, s_{[0,t+1]}, \dots, s_{[0,t_s]}$$

where $s_{[0,j]}$ is the subsequence of s in the interval $[0, j]$ and t_s is the length of the sequence s .

As in the case of sequences, causality is not sufficient when considering structured domains where the computational task requires complete contextual information, or, more in general, when there is no knowledge supporting the causality assumption. For instance, non-causal models can be useful when dealing with structured data where the meaning of substructures depends from the context in which they are found, i.e., in which position within a larger structure the given

substructure does occur. The challenge is therefore to study the possibility to process structures by a Recursive Neural Network model, relaxing the causality assumption.

In the following we describe a *Contextual Recursive Cascade Correlation* for directed acyclic structures (CRCC), based on an extension of the Recursive Cascade Correlation model¹ [1] (in this paper referred to as RCC), which constitutes, at the best of our knowledge, the first recursive neural model able to exploit the context in structured domains. Specifically, CRCC inherits and extends to structured domains the basic idea exploited in BRCC [9] and discussed above, i.e., contextual information stored in frozen hidden units is used by the pool of candidate hidden units to reduce the training error. Notice that the proposed model significantly differs from BRNN. In fact, first of all BRNN is only defined for sequences while our model can be applied to directed acyclic graphs. Moreover, BRNN generates two independent sets of hidden state variables, one taking information from the 'past' and one from the 'future', which are then fused by a set of top level state variables, while we exploit the fact that RCC has a reduced recurrence due to the freezing of intermediate hidden state variables. In fact, in our model every new inserted hidden state variable, besides to exploit the 'past' from all the inserted hidden state variables, can use also the information on the 'future' of the frozen hidden state variables, thus producing new hidden state variables that include both the 'past' and some information from the 'future'.

In this paper, we formally show that CRCC can compute contextual structural transductions which cannot be computed by RCC. Moreover, we demonstrate that some causal supersource transductions which cannot be computed by RCC, can be computed by CRCC, which on the other hand is able to compute all the transductions that can be computed by RCC. We are also able to formally elucidate how the "shape" (i.e., which state variables are accessed) of the contextual information evolves with the addition of hidden units.

Experimental results on controlled sequences and on a real-world task involving chemical structures confirm that CRCC is basically equivalent to RCC when considering a fully causal prediction task, while it is superior when considering contextual transductions (that RCC cannot compute) or prediction tasks where no information about the validity of the causality assumption is available.

The CRCC model was first proposed in [11], where however, no theoretical analysis was reported, as well as no extended experimental comparison versus the RCC model was performed.

II. STRUCTURED DOMAINS

Given a DAG (directed acyclic graph) \mathcal{D} we denote the vertexes set with $\text{vert}(\mathcal{D})$ and the edges set with $\text{edg}(\mathcal{D})$, where the edges are ordered couples of vertices. Given a vertex $v \in \text{vert}(\mathcal{D})$ we denote the set of edges entering and leaving from v with $\text{edg}(v)$. Moreover, we define:

- $\text{out_set}(v) = \{u | (v, u) \in \text{edg}(v)\}$;

¹The Recursive Cascade Correlation model is a generalization of the Recurrent Cascade Correlation model able to deal with structured information.

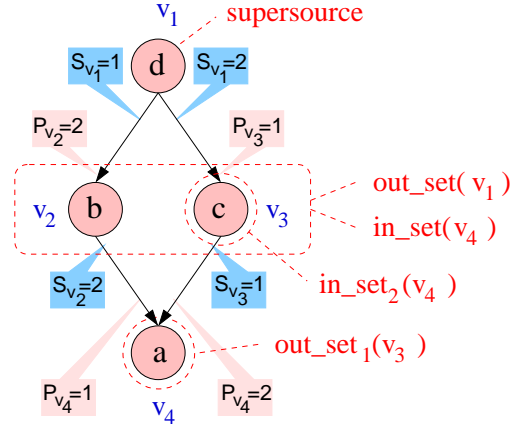


Fig. 1. Example of DPAG where some notations are shown. Specifically, $\text{out_set}(v_1) = \text{in_set}(v_4) = \{v_2, v_3\}$; $\text{out_set}_1(v_3) = v_4$, $\text{in_set}_2(v_4) = v_3$; $\text{in_deg}(v_4) = 2$, $\text{out_deg}(v_4) = 0$, $\text{in_deg}(v_1) = 0$, $\text{out_deg}(v_1) = 2$. We have also shown the functions $P_v()$ and $S_v()$ for each v . Here, in order to avoid cluttering of the picture, the numerical labels are represented by letters inside vertexes. The supersource of the DPAG is v_1 .

- $\text{in_set}(v) = \{u | (u, v) \in \text{edg}(v)\}$;
- $\text{out_deg}(v) = |\text{out_set}(v)|$ (outdegree);
- $\text{in_deg}(v) = |\text{in_set}(v)|$ (indegree).

In this paper we assume that instances in the learning domain are DPAGs (directed positional acyclic graphs) with bounded outdegree out and indegree in . An instance of DPAG is a DAG where we assume that for each vertex $v \in \text{vert}(\mathcal{D})$, two injective functions $P_v : \text{edg}(v) \rightarrow [1, 2, \dots, in]$ and $S_v : \text{edg}(v) \rightarrow [1, 2, \dots, out]$ are defined on the edges entering and leaving from v . In this way, a positional index is assigned to each entering and leaving edge from a node v . Moreover, we define $\forall u \in \text{vert}(\mathcal{D})$

$$\forall j \in [1, \dots, in]$$

$$\text{in_set}_j(u) = \begin{cases} v & \text{if } \exists v \in \text{vert}(\mathcal{D}) | P_u((v, u)) = j \\ nil & \text{otherwise} \end{cases}$$

$$\forall j \in [1, \dots, out]$$

$$\text{out_set}_j(u) = \begin{cases} v & \text{if } \exists v \in \text{vert}(\mathcal{D}) | S_u((u, v)) = j \\ nil & \text{otherwise} \end{cases}$$

We shall require the DPAGs to possess a supersource², i.e. a vertex $s \in \text{vert}(\mathcal{D})$ such that every vertex in $\text{vert}(\mathcal{D})$ can be reached by a directed path starting from s . With $\text{dist}(u, v)$, where $u, v \in \mathcal{D}$, we denote the shortest (directed) path in \mathcal{D} from the vertex u to the vertex v .

Moreover, vertices are labeled by vectors which either represent numerical or categorical variables. With $l(v)$ we denote the label associated to v , and with $l_i(v)$ the i -th element of the label. Examples of notation are shown in Fig. 1.

In the following, we shall denote by $\mathcal{L}^{\#(in, out)}$ the class of DPAGs with labels in the set \mathcal{L} , maximum indegree in and maximum outdegree out . In this paper we are interested in neural networks for the processing of structured domains.

²If no supersource is present, a new vertex connected to all the vertexes of the graph with null *indegree* can be added.

Specifically, we consider IO-isomorph transductions as defined in [4]. A structural transduction $\mathcal{T}(\cdot)$ is said to be *IO-isomorph* if

$$\text{skel}(\mathcal{T}(\mathcal{D})) = \text{skel}(\mathcal{D}) \quad \forall \mathcal{D} \in \mathbf{L}^{\#(in,out)},$$

where $\text{skel}(\mathcal{D})$ is the skeleton of \mathcal{D} obtained by ignoring all vertexes labels. It is well known that Recursive Neural Networks [3], which include the Recursive Cascade Correlation model, can only implement *causal* IO-isomorph transductions, i.e., the output computed at vertex v only depends on the information (i.e., labels and structural information) stored in v and descendants of v . Moreover, if both causality and stationarity are assumed, any causal IO-isomorph transduction can be described by a *supersource* transduction, i.e., a transduction that computes an output only for the supersource of the input DPAG. It is worth noting that under these conditions, a Recursive Neural Network cannot compute any causal transduction [4]. In the following we discuss a new model able to extend the class of functions which can be computed on structured domains.

III. CONTEXTUAL RECURSIVE MODEL

Recursive Neural Networks [2], [3] possess, in principle, the ability to memorize “past” information to perform structural mappings. The state transition function $\tau(\cdot)$ and the output function $g(\cdot)$, in this case, prescribe how the state variable, or better the state vector $\mathbf{x}(v)$ associated to a vertex v is used to obtain the state and output vectors corresponding to other vertexes, respectively. Specifically, given a state vector $\mathbf{x}(v) \equiv [x_1(v), \dots, x_m(v)]^t$, we define extended shift operators $q_j^{-1}x_k(v) \equiv x_k(\text{out_set}_j(v))$ and $q_j^{+1}x_k(v) \equiv x_k(\text{in_set}_j(v))$. If $\text{out_set}_j(v) = \text{nil}$ then $q_j^{-1}x_k(v) \equiv x_0$, the null state³. Similarly, if $\text{in_set}_j(v) = \text{nil}$ then $q_j^{+1}x_k(v) \equiv x_0$. Moreover, we define

$$\mathbf{q}^{-1}\mathbf{x}_k(v) = \begin{bmatrix} q_1^{-1}x_k(v) \\ \vdots \\ q_{out}^{-1}x_k(v) \end{bmatrix}, \quad (1)$$

$$\mathbf{q}^{+1}\mathbf{x}_k(v) = \begin{bmatrix} q_1^{+1}x_k(v) \\ \vdots \\ q_{in}^{+1}x_k(v) \end{bmatrix}, \quad (2)$$

and, given $e \in \{-1, +1\}$,

$$\mathbf{q}^e\mathbf{x}(v) = [\mathbf{q}^e x_1(v), \dots, \mathbf{q}^e x_m(v)]. \quad (3)$$

On the basis of these definitions, the mapping implemented by a Recursive Neural Network can be described by the following equations:

$$\begin{cases} \mathbf{x}(v) = \tau(\mathbf{l}(v), \mathbf{q}^{-1}\mathbf{x}(v)) \\ \mathbf{y}(v) = g(\mathbf{l}(v), \mathbf{x}(v)) \end{cases} \quad (4)$$

where $\mathbf{x}(v)$ is the network state associated to vertex v . This formulation, however, is based on a structural version of the *causality* assumption, i.e., the output $\mathbf{y}(v)$ of the network at vertex v only depends on descendants of v . Specifically, RCC

³In this paper, we assume $x_0 = 0$. Other assumptions can be considered, e.g. having a different null state for each j and missing entering/leaving edge.

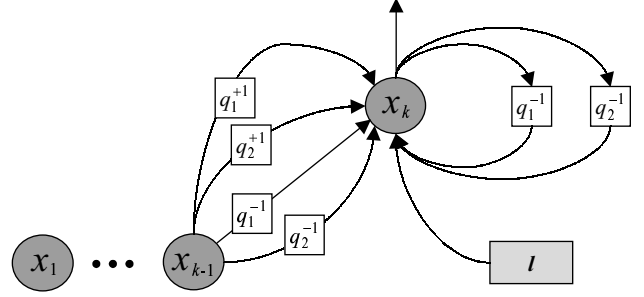


Fig. 2. Graphical model for x_k in CRCC. Only the functional dependencies for k and $k-1$, besides to the input, are shown explicitly. Here we assume $in = out = 2$.

equations, where we disregard direct connections between hidden units, can be written, for $j = 1, \dots, m$, as

$$x_j(v) = \tau_j(\mathbf{l}(v), \mathbf{q}^{-1}[x_1(v), \dots, x_j(v)]), \quad (5)$$

where $x_i(v)$ is the output of the i -th hidden unit in the network when processing vertex v . Since RCC is a constructive algorithm, training of a new hidden unit is based on already frozen units. Thus, when training hidden unit k , the state variables x_1, \dots, x_{k-1} for *all* the vertexes of all the DPAGs in the training set are already available, and can be used in the definition of x_k . This observation is very important since it yields to the realization that *contextual* information is already available in RCC, but it is not exploited.

Our proposal is to exploit this contextual information when training of a new hidden unit. Specifically, equations (5) can be expanded in a contextual fashion by using, where possible, the shift operator \mathbf{q}^{+1} :

$$x_1(v) = \tau_1(\mathbf{l}(v), \mathbf{q}^{-1}x_1(v)), \quad (6)$$

$$x_j(v) = \tau_j(\mathbf{l}(v), \mathbf{q}^{-1}[x_1(v), \dots, x_j(v)], \mathbf{q}^{+1}[x_1(v), \dots, x_{j-1}(v)]), \quad (7)$$

$$j = 2, \dots, m.$$

which constitute the equations for the proposed Contextual Recursive Cascade Correlation (CRCC). It should be noted that in eq. (7) the shift operator \mathbf{q}^{+1} cannot be applied to $x_j(v)$ since this would introduce a cyclic dependence in the definition of the variable, i.e., a dependence of $x_j(v)$ from the state variables of v 's parents which, however, via the shift operator \mathbf{q}^{-1} depend on the value of $x_j(v)$ itself. A graphical model for x_k in CRCC is shown in Fig. 2. On the basis of this graphical model, given a vertex v , a simple neural implementation should consider a neuron with three different types of input connections: *i*) connections associated to the input label, $\mathbf{l}(v)$; *ii*) connections associated to *all* the state variables (i.e., both already frozen hidden units and the current j -th hidden unit) of the children of the current vertex v , $\mathbf{q}^{-1}[x_1(v), \dots, x_j(v)]$; *iii*) connections associated to the state variables of the parents of v which are already *frozen* (i.e., the output of already frozen hidden units computed for the parents of v), $\mathbf{q}^{+1}[x_1(v), \dots, x_{j-1}(v)]$:

CRCC neuron

$$\begin{aligned}
x_j(v) &= f(\text{net}_j(v)) \\
&= f(\underbrace{\mathbf{w}_j^t}_i \mathbf{l}(v) + \sum_{i=1}^j \underbrace{\hat{\mathbf{w}}_{ji}^t}_{ii} \mathbf{q}^{-1} x_i(v) + \\
&\quad + \sum_{i=1}^{j-1} \underbrace{\hat{\mathbf{w}}_{ji}^t}_{iii} \mathbf{q}^{+1} x_i(v)),
\end{aligned}$$

where $f()$ is a sigmoidal function, e.g. $f(x) = \frac{1}{1+e^{-x}}$. Note that standard RCC does not have connections described in *iii*):

RCC neuron

$$\begin{aligned}
x_j(v) &= f(\text{net}_j(v)) \\
&= f(\underbrace{\mathbf{w}_j^t}_i \mathbf{l}(v) + \sum_{i=1}^j \underbrace{\hat{\mathbf{w}}_{ji}^t}_{ii} \mathbf{q}^{-1} x_i(v)).
\end{aligned}$$

A. Computational Properties of CRCC

Since CRCC exploits contextual information, it is important to fully understand the implications on the class of functions that the proposed model can compute. In the following, we give a definition of “contextual window” for a state variable and elucidate how the “shape” of the “contextual window” evolves with the addition of hidden units. A formal treatment of this issue is due, since understanding the effect of the introduction of the \mathbf{q}^{+1} operator in conjunction with the possibility to process graphs is not so straightforward as it may appear at a first look. For example, the intuition that, in a CRCC model with k hidden units, the context for a vertex v is given by all its descendants plus all the vertices that can be reached from v using a path with at most $k-1$ arcs followed in the opposite direction, is telling only a part of the truth, since to obtain the actual context the same argument must be recursively applied to all the descendants of v . An additional advantage of the formal treatment we suggest here is the derivation of a very compact expression characterizing the contextual window of a state variable. This is obtained by defining ad hoc operators that work on sets of vertices and state variables. Moreover, this formal treatment will allow us to prove that CRCC can compute functions which cannot be computed by RCC. Finally, it should be stressed that the formal approach used in the following is independent from the specific neural realization of eqs. (6) and (7). Thus, the same approach can be used for any graphical model defined on structured domains in order to formally determine the contextual window of a state variable.

In order to formalize the above concepts, let us define the operator \downarrow which applied to a subset of vertexes $V \subseteq \text{vert}(\mathcal{D})$ returns the union of V with the set of descendants of vertexes in V , i.e., $\downarrow V = V \cup \{u | v \in V \wedge \exists \text{path}(v, u)\}$. Moreover, let the set $\text{in_set}(\cdot)$ to be defined also when the argument is a subset of vertexes V , i.e., $\text{in_set}(V) = \bigcup_{v \in V} \text{in_set}(v)$ and let denote with $\text{in_set}^p(V)$ the repeated application of the function

for p times, i.e.,

$$\text{in_set}^p(V) \equiv \underbrace{\text{in_set}(\cdots(\text{in_set}(V))\cdots)}_{p \text{ times}}$$

Actually, we will use the repeated application of the in_set function composed with \downarrow , i.e.,

$$(\downarrow \text{in_set})^p(V) \equiv \downarrow \text{in_set}(\cdots(\downarrow \text{in_set}(V))\cdots)$$

Finally, with $x_j.V$ we refer to the set of state variables $\{x_j(v) | v \in V\}$.

Definition 1: Given a state variable $x_k(v)$ we define its context window, denoted $\mathcal{C}(x_k(v))$, as the set of *all* the state variables which (directly or indirectly) contribute to its determination.

For CRCC, it is possible to show the following result

Theorem 1: Given a DAG \mathcal{D} , for any vertex $v \in \text{vert}(\mathcal{D})$, and for any index $k \geq 2$, the following equation holds for CRCC

$$\mathcal{C}(x_k(v)) = \bigcup_{i=1}^{k-1} x_i \cdot (\downarrow \text{in_set})^{k-i}(\downarrow v) \cup x_k \cdot \downarrow \text{out_set}(v). \quad (8)$$

Proof: Here we present a sketch of the proof. For a complete and detailed proof see [12].

The proof can be obtained by first of all showing that, by induction on the partial order of vertexes, for any v , $\mathcal{C}(x_1(v)) = x_1 \cdot \downarrow \text{out_set}(v)$. As a corollary of that, given a DPAG \mathcal{D} , for any couple of vertexes $u, v \in \text{vert}(\mathcal{D})$ connected by a path from v to u then $\mathcal{C}(x_1(u)) \subseteq \mathcal{C}(x_1(v))$. These results are used as basis for showing, again by induction on the partial order of vertexes, that for any v , $\mathcal{C}(x_2(v)) = x_1 \cdot \downarrow \text{in_set}(\downarrow v) \cup x_2 \cdot \downarrow \text{out_set}(v)$ and consequently, for any vertex $v \in \text{vert}(\mathcal{D})$ and for any $k \geq 2$, it holds $\mathcal{C}(x_k(v)) \supseteq \mathcal{C}(x_{k-1}(v))$. Using the first three results as base cases and the last one as induction rule, eq. (8) can be proven by induction on the partial order of vertexes, and on the order of indexes of variables. ■

This theorem clearly shows that the introduction of each new hidden unit in CRCC increases of one “step” the size of the context in the direction of the “future”. This expansion of the context can easily be visualized when considering a temporal sequence, which in our framework is represented as a list where the last element is denoted by v_1 and element v_t corresponds to the item occurring at time t . In this case, the context becomes

$$\mathcal{C}(x_k(v_t)) = \bigcup_{i=1}^{k-1} x_i \cdot \downarrow v_{t+k-i} \cup x_k \cdot \downarrow v_{t-1} \quad (9)$$

An example of visualization of eq. (9) is reported in Fig. 3 for $k=3$ and $t=2$. When considering a tree, the context grows (via the in_set function) including all the subtrees (because of \downarrow) rooted in vertexes met along the (inverse) path between the current vertex and the root (i.e., supersource)

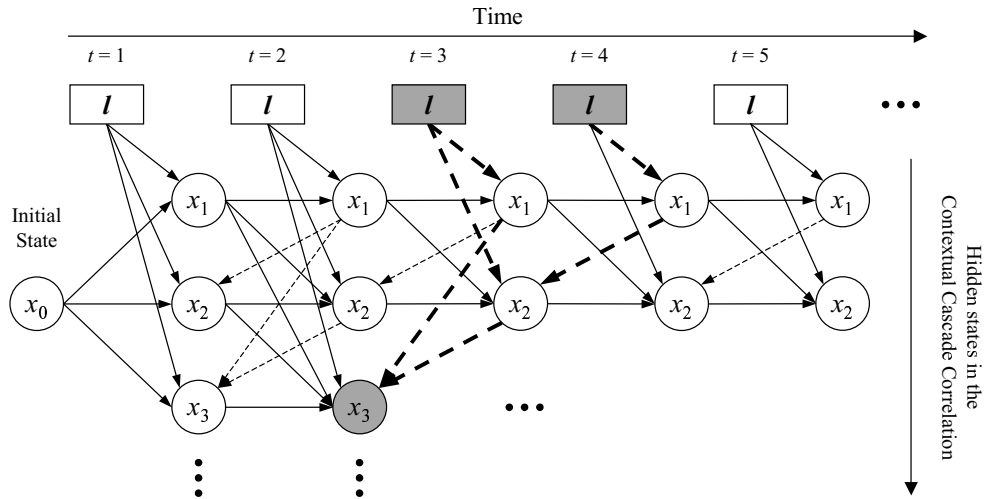


Fig. 3. Unrolling of the graphical model of our CRCC when considering a temporal sequence. State variables are shown up to $k = 3$. As an example, to reveal contextual information at x_3 , links which convey information about the future to x_3 at time $t = 2$ (i.e., $t = 3$ and $t = 4$ in the example) are shown in bold.

of the tree. An example of expansion of the context for a tree is given in Figure 4, where we show how adding new hidden units to the CRCC network leads to an increase of the “context window” associated to each vertex v . Specifically, the shown example focuses on the state computation of the vertex labeled “d” in the input tree, and describes for it, in a pictorial way through boxes, the functional dependences introduced by any new hidden unit inserted in the network. Unit 1 (see eq. (6)) implements only causal computation. After adding unit 2, contextual information concerning the subtree rooted in the vertex labeled “g”, contributes to the state definition of the vertex labeled “d”. Finally, after adding unit 3, the context is extended to the whole tree. The growing of the context for a DPAG is a bit harder to understand. This is due to the fact that in a DPAG a vertex may have more than one entering edge. So the context grows (via the in_set function) including all the sub-DPAGs (because of \downarrow) rooted in vertexes met along all the (inverse) paths between the current vertex and the supersource. Moreover, paths reaching descendants of vertexes included in the current context must be taken into account when considering the new context obtained by adding a new hidden unit. An example of how the context grows for a DPAG is shown in Figure 5.

In general, for CRCC, the evolution of the context with respect to the addition of hidden units is characterized by the following property.

Proposition 1: Given a vertex v in a DPAG \mathcal{D} with supersource s , such that $\text{dist}(s, v) = d$, the contexts $\mathcal{C}(x_h(v))$, with

$h > d$ involve all the vertexes of \mathcal{D} .

Proof: According to eq. (8), when computing $\mathcal{C}(x_{d+1}(v))$, in_set is recursively applied d times. Thus the shortest path from s to v is fully followed in a backward fashion starting from v , so that $x_1(s)$ is included in $\mathcal{C}(x_{d+1}(v))$. Moreover, since $x_1(s)$ is included in $\mathcal{C}(x_{d+1}(v))$, by definition of eq. (8), also the state variables $x_1(u)$ for each $u \in \text{vert}(\mathcal{D})$ are included in $\mathcal{C}(x_{d+1}(v))$. The statement follows from the fact that $\mathcal{C}(x_{d+1}(v)) \subset \mathcal{C}(x_h(v))$. ■

Moreover, when considering a target function that for each vertex v depends on the whole structure, the following proposition suggests that such information becomes available to the CRCC model.

Proposition 2: Given a DPAG \mathcal{D} there exists a finite number h of hidden units such that for each $v \in \text{vert}(\mathcal{D})$ the context $\mathcal{C}(x_h(v))$ involves all the vertexes of \mathcal{D} . In particular, given $r = \max_{v \in \text{vert}(\mathcal{D})} \text{dist}(s, v)$, any $h > r$ satisfies the proposition.

Proof: Let consider $h = r + 1$. The proposition follows immediately by the application of Proposition 1 for each v since $h > \text{dist}(s, v)$. ■

Note that, differently from CRCC, the context for RCC is characterized by the following

Theorem 2: Given a DAG \mathcal{D} , for any vertex $v \in \text{vert}(\mathcal{D})$, and for any index $k \geq 1$, the following equation holds for

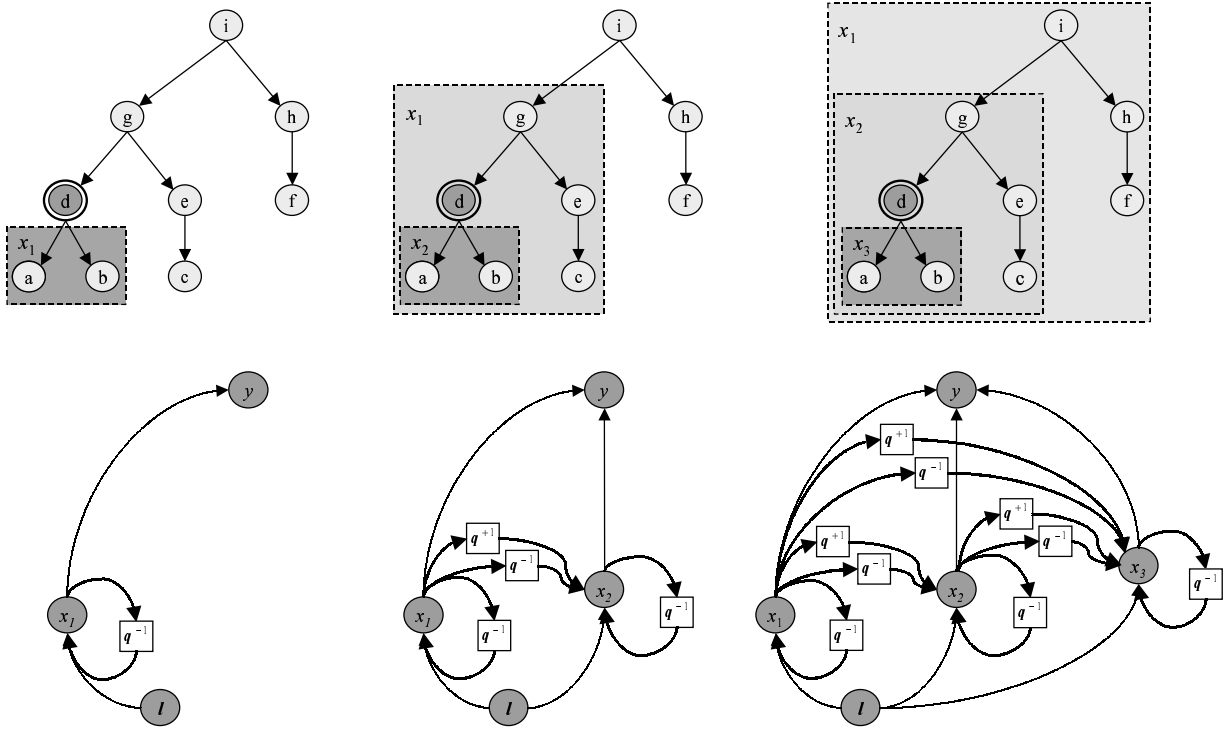


Fig. 4. Evolution of the “context window” for the vertex labeled “d” in the input tree with the growing of the network, shown at the lower part of the figure via the evolution of the graphical model of CRCC. The factorization of the context, as shown in eq. (8), is visualized by boxes referring to corresponding state variables.

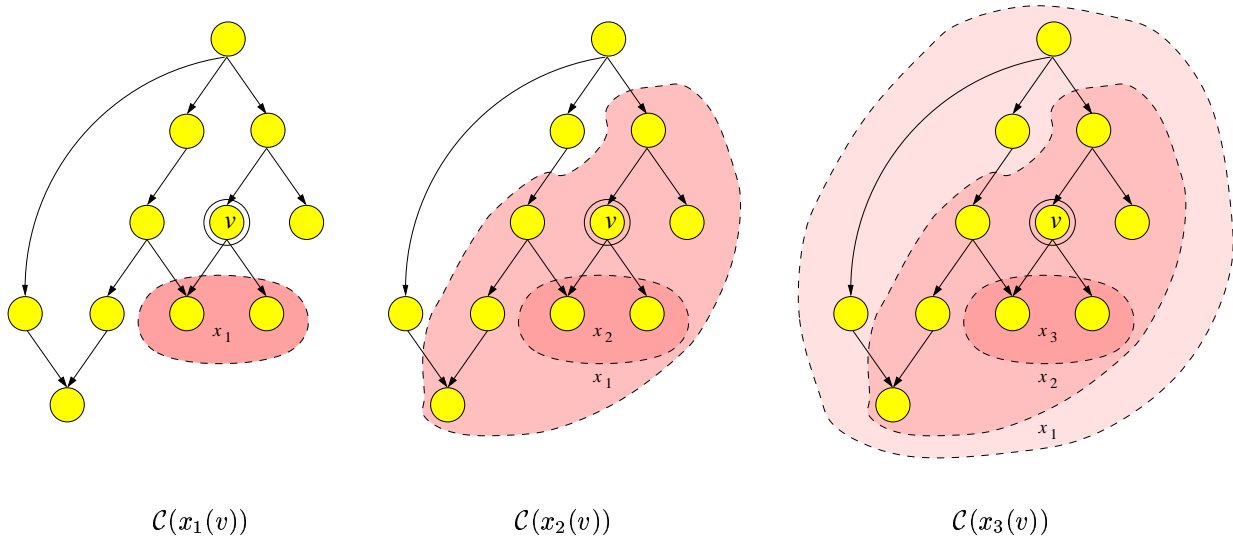


Fig. 5. Example, for a DPAG, of the evolution of the “context windows” for the state variables referring to vertex v . Notice how the context of state variable x_2 includes also vertexes which are not descendants of v , and how the context of state variable x_3 includes the full graph.

RCC

$$\mathcal{C}_{RCC}(x_k(v)) = \bigcup_{i=1}^k x_i \cdot \downarrow \text{out_set}(v) \quad (10)$$

Proof: The proof is readily obtained by observing that, because of the fully causal assumption on the children that holds for RCC, the state variable of vertex v only depends on

state variables of vertexes that are descendants of v . ■

Note that, since `in_set` does not occur in the formula, Proposition 1 and Proposition 2 cannot hold. In fact, RCC can only use information about the descendants of a vertex v , and for this reason it will never be able to include other vertexes in any context of v , while CRCC can do it. From a computational point of view this implies that RCC will never be able to compute a function whose output for a given vertex v depends on vertexes of the input structure that are not descendants of v .

In addition to that, focusing the attention to supersource transductions, the following result holds

Lemma 1: There exist two distinct DPAGs, \mathcal{D}_1 with supersource s_1 and \mathcal{D}_2 with supersource s_2 , such that the state vectors $\mathbf{x}^{RCC}(s_1)$ and $\mathbf{x}^{RCC}(s_2)$ computed by RCC are identical, while the state vectors $\mathbf{x}(s_1)$ and $\mathbf{x}(s_2)$ computed by CRCC are different.

Proof: Let consider the two graphs in Fig. 6, where $s_1 \equiv v_1$ and $s_2 \equiv v_5$. We show that while RCC returns, regardless of the state vector dimension k , $\mathbf{x}^{RCC}(s_1) = \mathbf{x}^{RCC}(s_2)$, CRCC is able to produce different state vectors $\mathbf{x}(s_1)$ and $\mathbf{x}(s_2)$ as soon as the dimension of the state vector is higher than 1, i.e., $k \geq 2$. Specifically, let consider vertexes v_4 , v_8 , and v_9 (see Fig. 6). For $k \geq 1$, since the label attached to vertexes v_4 , v_8 , and v_9 is the same (“a”), we have

$$\mathbf{x}_k^{RCC}(v_4) = \mathbf{x}_k^{RCC}(v_8) = \mathbf{x}_k^{RCC}(v_9) = \mathbf{x}_a,$$

and

$$\mathcal{C}_{RCC}(x_k(v_4)) = \mathcal{C}_{RCC}(x_k(v_8)) = \mathcal{C}_{RCC}(x_k(v_9)) = \{x_0\}.$$

Because of that and eq. (5), for any value of k we have $\mathbf{x}^{RCC}(s_1) = \mathbf{x}^{RCC}(s_2)$. On the contrary, for CRCC this is not true since, e.g. $k = 2$, applying Theorem 1 we obtain:

$$\begin{aligned} \mathcal{C}(x_2(v_4)) &= \{x_1(v_2), x_1(v_3), x_1(v_4), x_0\} \\ \mathcal{C}(x_2(v_8)) &= \{x_1(v_6), x_1(v_8), x_0\} \\ \mathcal{C}(x_2(v_9)) &= \{x_1(v_7), x_1(v_9), x_0\} \end{aligned}$$

and, even if $x_1(v_4) = x_1(v_8) = x_1(v_9)$ (because of the same label “a”), in general we have that $x_1(v_2) \neq x_1(v_3)$ and $x_1(v_6) \neq x_1(v_7)$ (since vertexes v_2 and v_3 , as well as v_6 and v_7 , have different labels), which implies

$$\mathcal{C}(x_2(v_4)) \neq \mathcal{C}(x_2(v_8)) \neq \mathcal{C}(x_2(v_9)).$$

Thus, because of that and eq. (7), there exists a CRCC network able to compute different state vectors for v_4 , v_8 , v_9 , and consequently also to produce state vectors $\mathbf{x}(s_1)$ and $\mathbf{x}(s_2)$ that are different. ■

Note that, as seen in the proof of Lemma 1, CRCC is able to differentiate (to produce state variables with different values) between the two vertexes labeled “a” in the right hand side graph of Fig. 6. This is due to the fact that they have a different parent, thus their contexts are different.

The above Lemma can be used to state the following

Theorem 3: The class of functions which can be computed by RCC is properly included in the class of functions which can be computed by CRCC.

Proof: The class of functions computed by RCC is included in the class of functions computed by CRCC since eq. (7) can be reduced to eq. (5) by considering functions $\tau_j(\cdot)$ which do not consider inputs $\mathbf{q}^{+1}x_1(v), \dots, \mathbf{q}^{+1}x_{j-1}(v)$. Moreover, because of Lemma 1, there exist at least one supersource transduction which cannot be computed by RCC, because it cannot distinguish between the supersource states of at least two input DPAGs, while CRCC can distinguish them and compute such function. ■

Note that, because of Proposition 1 and Proposition 2, CRCC can also compute transductions that are not strictly causal. Examples of such functions are given in Section V, where we show that CRCC is able both to compute and learn them.

Summarizing, due to the addition of new connections carrying information from the context of each vertex, CRCC allows to consider, with respect to causal models, the following extension to the treatable classes of target functions:

- extension to *contextual* IO-isomorphic transductions, including the cases where the desired output for a given vertex v depends on vertexes of the input structure (sequences, tree or DPAG) that are not descendants of v (future dependencies) (e.g. the desired response for each vertex depending on the whole structure);
- extension to the class of supersource transductions that involve DPAGs that cannot be computed by causal models;

while supporting all the function computable by RCC models.

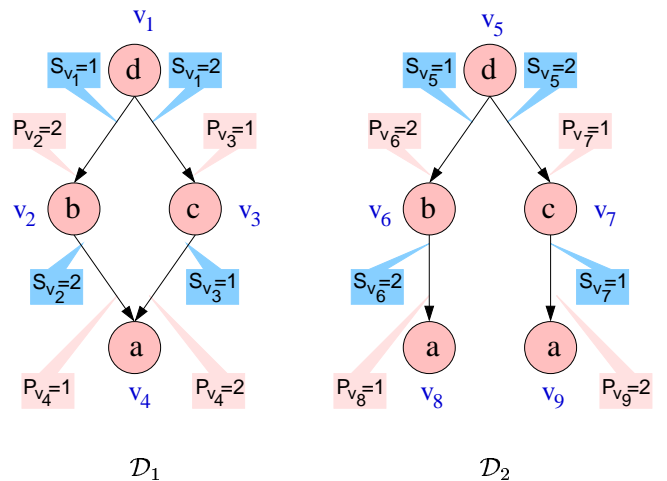


Fig. 6. Two graphs for which RCC produces state variables with identical values for each value of k , while CRCC is able to produce state variables with different values for $k \geq 2$. In particular, CRCC returns different state variables for the two vertexes labeled “a” in the right side graph since their `in_set` is different, i.e., they have different context.

IV. CONTEXTUAL RECURSIVE CASCADE CORRELATION

Concerning the neural realization of CRCC, the state vector over the current vertex v can be computed as⁴

$$\mathbf{x}(v) = \mathbf{F}(\mathbf{net}(v)) = [f(\mathbf{net}_1(v)), \dots, f(\mathbf{net}_m(v))]^t \quad (11)$$

$$\mathbf{net}(v) = \begin{bmatrix} \mathbf{w}_1^t \\ \mathbf{w}_2^t \\ \mathbf{w}_3^t \\ \vdots \\ \mathbf{w}_m^t \end{bmatrix} l(v) + \quad (12)$$

$$\begin{bmatrix} \hat{\mathbf{w}}_{11}^t \mathbf{q}^{-1} x_1(v) \\ \sum_{i=1}^2 \hat{\mathbf{w}}_{2i}^t \mathbf{q}^{-1} x_i(v) \\ \sum_{i=1}^3 \hat{\mathbf{w}}_{3i}^t \mathbf{q}^{-1} x_i(v) \\ \vdots \\ \sum_{i=1}^m \hat{\mathbf{w}}_{mi}^t \mathbf{q}^{-1} x_i(v) \end{bmatrix} + \quad (13)$$

$$\begin{bmatrix} 0 \\ \tilde{\mathbf{w}}_{21}^t \mathbf{q}^{+1} x_1(v) \\ \sum_{i=1}^2 \tilde{\mathbf{w}}_{3i}^t \mathbf{q}^{+1} x_i(v) \\ \vdots \\ \sum_{i=1}^{m-1} \tilde{\mathbf{w}}_{mi}^t \mathbf{q}^{+1} x_i(v) \end{bmatrix} \quad (14)$$

where $f(\cdot)$ is a sigmoidal function; \mathbf{w}_k is the label weight vector associated to the k -th hidden unit; $\hat{\mathbf{w}}_{ki}$, with $1 \leq i \leq k \leq m$, are the k weight vectors associated with the k -th hidden unit, each one associated to the ‘‘past’’ information arriving through the outgoing edges of the current vertex from the i -th (frozen, for $i < k$) hidden unit; $\tilde{\mathbf{w}}_{ki}$, with $1 \leq i < k \leq m$, are the $k - 1$ weight vectors associated to the k -th hidden unit, each one associated to the ‘‘future’’ information arriving through the ingoing edges of the current vertex from the i -th (frozen, for $i < k$) hidden unit.

Note that the first component (12) corresponds to the contribution of the ‘‘present’’ information, i.e., the label attached to v , the second component (13) corresponds to the contribution of the ‘‘past’’ information coming from descendants of v , while the last component (14) corresponds to the contribution of the ‘‘future’’ information coming from the subgraphs with supersource in the set $\text{in_set}(v)$. Given an input structure, the network output function $g(\cdot)$ (see Eq. 4) is implemented by one or more standard neurons

$$\mathbf{y}(v) = \mathbf{F}(\mathbf{A}l(v) + \mathbf{B}\mathbf{x}(v)) \quad (15)$$

where \mathbf{A} is the output label weight matrix, and \mathbf{B} is the output state weight matrix, which is increased in size each time a new hidden unit is added.

Learning is performed as in standard Cascade Correlation by interleaving the minimization of the total error function (LMS) by a simple backpropagation training of the output layer, and the maximization of the (non-normalized) correlation, i.e. the covariance, of the new inserted hidden unit k with the residual

error:

$$S = \sum_u \left| \sum_v (x_k(v) - \bar{x}_k)(E_u(v) - \bar{E}_u) \right| \quad (16)$$

where u spans over the output units, v spans over the vertexes of all input structures for which a target is defined, \bar{x}_k is the mean output of the current unit, $E_u(v)$ is the residual error of the output unit u for vertex v , and \bar{E}_u is the mean residual error of the output unit u .

The weight variation is then computed by the standard gradient ascent approach, deriving equation (16) with respect to the desired weight:

$$\Delta w_{ki} = \eta \frac{\partial S}{\partial w_{ki}} = \eta \sum_u \sigma_u \sum_v (E_u(v) - \bar{E}_u) \frac{\partial x_k(v)}{\partial w_{ki}} \quad (17)$$

where σ_u is the sign of the correlation between the output of the current hidden unit and the residual error of the output unit u .

Applying the RTRL algorithm approach as described in [13] we can determine the derivative of the output of the current hidden unit as follows:

$$\frac{\partial x_k(v)}{\partial \mathbf{w}_k} = f' \cdot \left(l(v) + \frac{\partial \mathbf{q}^{-1} x_k(v)}{\partial \mathbf{w}_k} \hat{\mathbf{w}}_{kk} \right) \quad (18)$$

$$\frac{\partial x_k(v)}{\partial \hat{\mathbf{w}}_{ki}} = f' \cdot \left(\mathbf{q}^{-1} x_i(v) + \frac{\partial \mathbf{q}^{-1} x_k(v)}{\partial \hat{\mathbf{w}}_{ki}} \hat{\mathbf{w}}_{ki} \right) \quad (19)$$

$$\frac{\partial x_k(v)}{\partial \tilde{\mathbf{w}}_{ki}} = f' \cdot \left(\mathbf{q}^{+1} x_i(v) + \frac{\partial \mathbf{q}^{-1} x_k(v)}{\partial \tilde{\mathbf{w}}_{ki}} \hat{\mathbf{w}}_{ki} \right) \quad (20)$$

where f' is the first derivative of $f(\cdot)$. Note that equations (18), and (19) are the same used in standard RCC for structured data [1], while equation (20) is added so to include also contextual (‘‘future’’ in sequences) information from frozen units. The above equations are recurrent and can be computed by observing that for all the leaves of the structured data (all vertexes with null outdegree) eq. (18) becomes $\frac{\partial x_k(v)}{\partial \mathbf{w}} = f' l(v)$, the derivatives for eq. (19) are null, and eq. (20) reduces to $\frac{\partial x_k(v)}{\partial \tilde{\mathbf{w}}_{ki}} = f' \mathbf{q}^{+1} x_i(v)$. Consequently, we only need to store the output values of the unit and its derivatives for each component of the structure.

A. Special Cases and Extensions

When considering a structured domain with maximum in-degree and outdegree equal to 1, e.g. temporal sequences, eqs. (1)-(2) reduce to the following:

$$\mathbf{q}^{-1} x(t) = x(t-1), \quad (21)$$

$$\mathbf{q}^{+1} x(t) = x(t+1). \quad (22)$$

In this framework, RCC reduces to the Recurrent Cascade Correlation model [10], and the CRCC model becomes the *Bi-causal Recurrent Cascade Correlation* [9], where eq. (7) reduces to

$$x_j(t) = \tau_j(l(t), x_1(t-1), \dots, x_j(t-1), x_1(t+1), \dots, x_{j-1}(t+1)). \quad (23)$$

⁴Notwithstanding the vectorial representation, here the computation has to be understood in a data-flow fashion, i.e., the component i -th of the vector can be computed only when the component $i - 1$ -th has already been computed.

On the other hand, the architecture described in Section III can of course be extended in several ways. For example, we can consider the composition of shift operators:

$$\mathbf{q}^{-p} = \underbrace{\mathbf{q}^{-1}\mathbf{q}^{-1}\dots\mathbf{q}^{-1}}_{p \text{ times}}$$

and

$$\mathbf{q}^{+p} = \underbrace{\mathbf{q}^{+1}\mathbf{q}^{+1}\dots\mathbf{q}^{+1}}_{p \text{ times}},$$

where, given $e \in \{-1, +1\}$, we have the following rule

$$\mathbf{q}^e \begin{bmatrix} x_{11} & \dots & x_{1t} \\ \vdots & & \vdots \\ x_{r1} & \dots & x_{rt} \end{bmatrix} = \begin{bmatrix} \mathbf{q}^e x_{11} & \dots & \mathbf{q}^e x_{1t} \\ \vdots & & \vdots \\ \mathbf{q}^e x_{r1} & \dots & \mathbf{q}^e x_{rt} \end{bmatrix}.$$

So it is possible to extend eqs. (6) and (7) by introducing shift operators with $p > 1$ and/or $p < -1$, or combinations of them⁵. Moreover, when considering the neural realization, both $\tau(\cdot)$ and $g(\cdot)$ can be implemented by a multilayer network instead of a single layer of neurons. For all of these extensions it is not difficult to see that gradients can be easily computed and new suitable learning rules can be devised.

V. EXPERIMENTAL RESULTS

In the following we report the results obtained with the Contextual Recursive Cascade Correlation model applied to regression tasks in different structured domains involving sequences and trees, respectively. The main aim of Section III-A was to give theoretical support to the proposed contextual method, showing that CRCC can compute contextual functions that are not computable by RCC. Experimental results, however, besides to verify the theory, allow us also to study CRCC beyond theoretical results: i.e., how does CRCC behaves in tasks where the reliability of a causality assumption is unknown (for instance in the case of supersource transductions that both RCC and CRCC can in principle compute) ? Or, assuming that the causality assumption does hold, how inefficient the CRCC is when compared to RCC ?

For the sequence domain we have decided to use artificial data sets in order to have the control of the causality/contextual conditions in evaluating the ability of CRCC in learning contextual mappings. The aim of our experiments in this domain is to show that while RCC is unable to learn a contextual mapping, as expected from the theoretical results, CRCC can do it. Furthermore, through the use of carefully controlled experiments, we show that this ability of the CRCC does not impair the prediction ability of the model under strict causality conditions.

The structured domain involving trees, on the other hand, concerns a real-world problem in Chemistry involving supersource transductions, i.e., a Quantitative Structure Property Relationship (QSPR) analysis of alkanes. This supersource transduction can in principle be computed by both RCC and CRCC. In fact, the prediction of the property of an input tree is performed only after the whole tree has been processed by

the models, i.e., when processing the root (or supersource) of the tree. Thus, this experiment allows us to compare RCC and CRCC on a fair ground, since: *i)* RCC is able to generate different state vectors for distinct trees; *ii)* the general form of the target function is unknown and so it is not clear whether the causality assumption holds.

It should be stressed that this particular experiment is complementing the theoretical results that state the in principle inability of RCC to compute any contextual mapping, since it is not known, in this case, whether the prediction task needs contextual information or not.

In all the experiments described below, both for RCC and CRCC, we have adopted the following parametrization for the output function (one single linear output)

$$y(v) = \mathbf{z}^t \mathbf{x}(v).$$

A. Learning Contextual Mappings for Sequences

For this domain, we have considered regression problems. Different sets of randomly generated artificial sequences have been produced. The training sets are composed of 200 sequences while the test sets are composed of 100 sequences. Of course, in this case the maximum indegree and outdegree is 1. The sequences, of length between 5 and 20, are composed of symbols in the alphabet $\mathcal{A} = \{a, \dots, j\}$. Each symbol is selected according to a uniform distribution over the alphabet and it is coded as a 10-bit string, with one specific bit turned on (+1) and all others turned off (-1). This representation of symbols guarantees that no a priori metrics is imposed on them. Moreover, in order to define the target, a function $v : \mathcal{A} \rightarrow \{0, 0.1, \dots, 0.9\}$ is defined (i.e., $v(a) = 0, \dots, v(j) = 0.9$). Notice that the mapping of symbols in real numbers for generating the target function is not critical for the learning task since the input symbols do not have any bias versus the induced order, and anyway it is just a way to produce a regression task on which to compare CRCC versus RCC.

Different prediction tasks were obtained by defining different target functions for each element of a sequence.

The first target function, strictly dependent on the next position in the sequence, is defined as in the following

$$\text{target}_{1f}(t) = v(s(t+1)), \quad (24)$$

where $s(t+1)$ returns the sequence element in position $t+1$.

For comparison with RCC we have also used the following causal target function (which depends only on the past element):

$$\text{target}_{1p}(t) = v(s(t-1)). \quad (25)$$

Other target functions involving the average on a window of size 4 have been used:

$$\text{target}_{4f}(t) = \frac{\sum_{j=0}^3 v(s(t+j))}{4}, \quad (26)$$

$$\text{target}_{4p}(t) = \frac{\sum_{j=0}^3 v(s(t-j))}{4}. \quad (27)$$

Finally, we have defined a moving average target over the future:

$$\text{target}_{ma}(t) = \frac{\text{target}_{ma}(t+1) + v(s(t))}{2}. \quad (28)$$

⁵Please, note that \mathbf{q}^{-1} and \mathbf{q}^{+1} are not commutative.

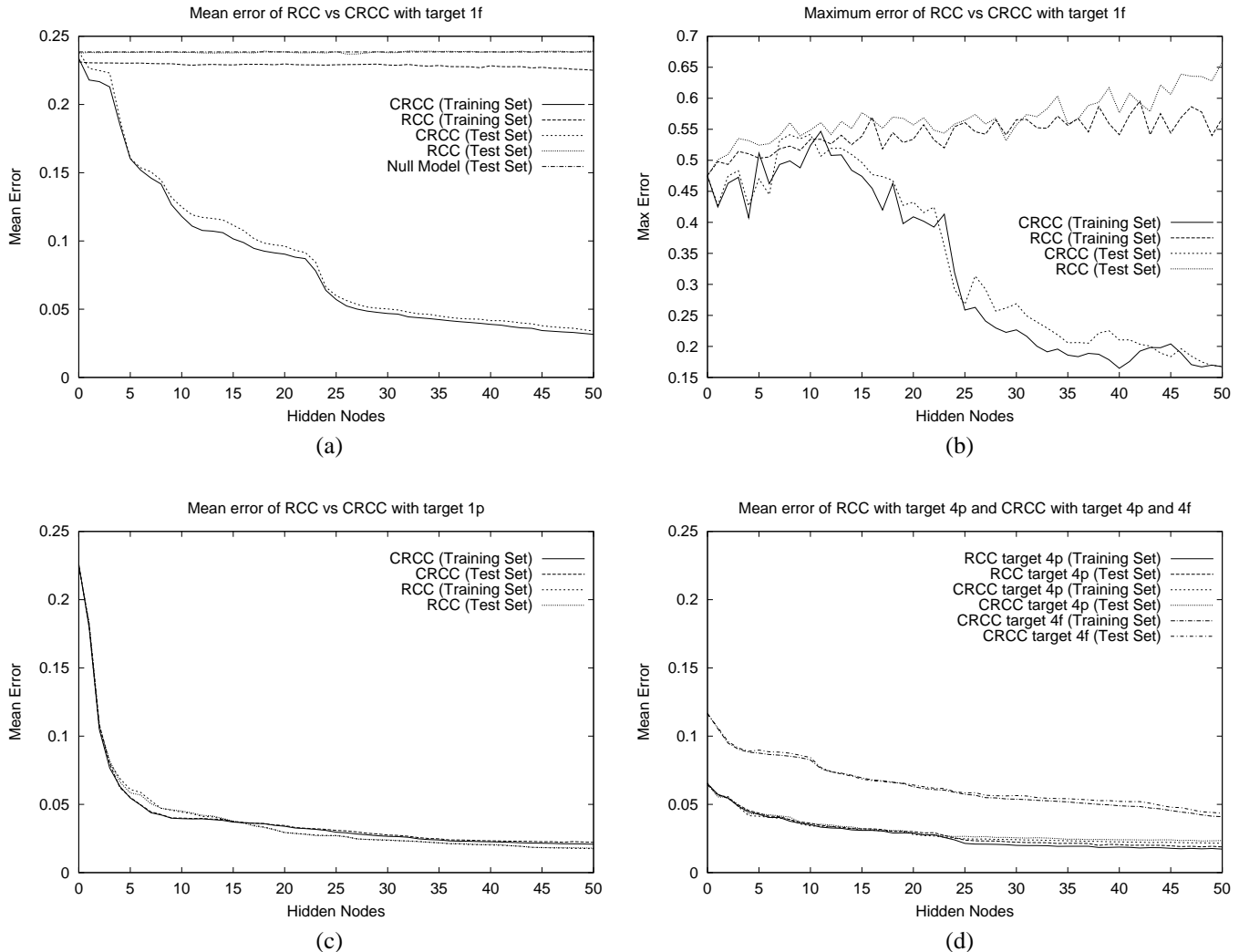


Fig. 7. Top: mean (a) and maximum (b) error of CRCC and RCC for $target_{1f}$. Note that the mean error of RCC is near the mean error of the null model, and the maximum error diverges. Bottom: mean error for RCC and CRCC for $target_{1p}$ (c), $target_{4p}$ and $target_{4f}$ (d).

Clearly, both $target_{1p}$ and $target_{4p}$ define computational tasks for which the causality assumption fully holds. On the other hand, the causality assumption does not hold when using $target_{1f}$, $target_{4f}$, and $target_{ma}$, which can be considered “contextual” functions.

It should be stressed that the role of these experiments is to show that contextual target functions cannot be learned by RCC, while CRCC can learn them. Moreover, even when RCC can learn the target functions, CRCC can do it as well without losing efficiency. Of course, the proposed target functions could be learned without effort by a feedforward network with a suitable choice of the size of the input window, however this is not the main point in these experiments, since the use of a recurrent model is justified when no a priori information about the size of the input window is available.

We performed several training trials with all the above defined target functions. Examples of typical error curves observed for CRCC and RCC, for each prediction task, are reported.

An example of the results obtained by CRCC and RCC

over $target_{1f}$ are given in Figure 7(a-b). The performance of a theoretical null model⁶ for the test set is shown as well. Note that, as expected, the RCC is not able to improve over the null model. The difficulty of RCC to deal with the prediction task is also evident from the increase in the maximum error corresponding to the increase of the number of hidden units into the network (see Figure 7(b)). On the contrary, CRCC is able to decrease the maximum error along with the increase in the number of hidden units.

As shown in Figure 7(c), when considering $target_{1p}$ (i.e., the causality assumption holds) the results obtained by CRCC are comparable with those obtained by RCC. This shows that the CRCC’s ability to use contextual information does not impair the performance of the model under strict causal conditions. A confirmation of this behavior is given when experimenting with $target_{4p}$ (see Figure 7(d)).

Finally, CRCC seems to be able to cope well with longer dependencies in the future, as encoded in $target_{4f}$ (see

⁶The null model is obtained by computing the expected value for the target over the training set.

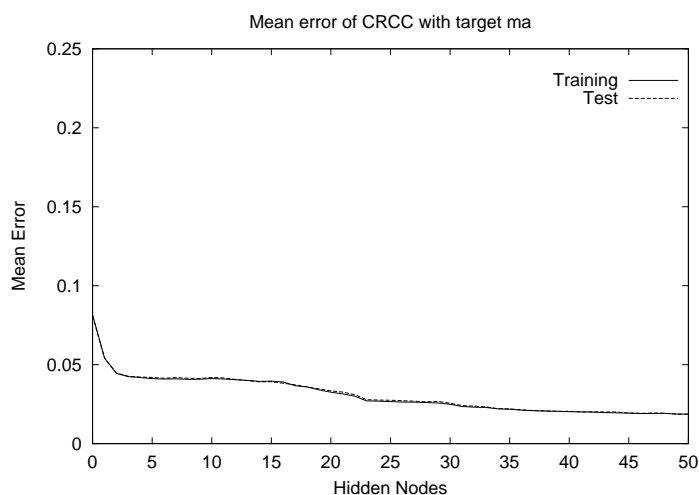


Fig. 8. Mean error for CRCC on $target_{ma}$.

Figure 7(d)), as well as with the moving average over the future, i.e., $target_{ma}$ (see Figure 8).

In all the experiments we let the training to insert many more hidden units than necessary for solving the regression problems, however, it can be noticed that no overfitting was observed. This is due to the adoption of a regularization strategy called *i-strategy* that is fully described in [14].

B. QSPR Analysis of Alkanes

Here we consider a regression problem on a structured domain involving chemical compounds represented as trees. The problem consists in the prediction of the boiling point for a group of acyclic hydrocarbons (alkanes). The boiling temperature of alkanes is frequently used as a benchmark property in testing Quantitative Structure-Property Relationship (QSPR) models. For this problem, the causal model (RCC) has been proved to be competitive with respect to *ad-hoc* techniques (see [14]). In fact, the results obtained by RCC compares favorably versus the approach proposed by Cherqaoui et al. [15]. They apply a multilayer feed-forward neural network to a vectorial representation of alkanes able to retain the structural information which is known to be relevant to the prediction of the boiling point.

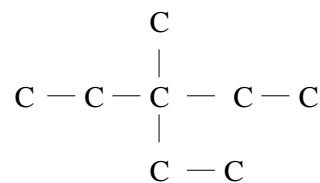
This task has been selected in order to have a direct comparison of the new approach (CRCC) versus the standard causal model (RCC) in a real-world application. Since the target property is related to global characteristics of the structures, such as the molecular size and the molecular shape, we believe that a model able to capture contextual information should improve the performance on this task.

Moreover, these experiments allow to investigate the coherence of the causality assumption, and the effect of its relaxation, on a real-world application.

The data set used here, which is taken from [14], is based on all the 150 alkanes with up to 10 carbon atoms (C_nH_{2n+2}).

It is well known that for this class of compounds, the prediction of the boiling point can be performed by disregarding the information about the hydrogen atoms. Hydrogens

Chemical Compound (hydrogens suppressed)



Representation by Rooted Tree

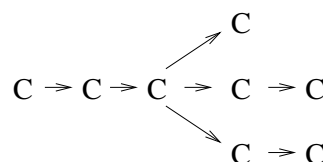


Fig. 9. Example of rooted-tree representation for an alkane (3-ethyl-3-methylpentane).

suppressed graphs of alkane molecules are trees. Carbon-hydrogens groups are associated with vertexes, and bonds between carbon atoms are represented by edges. In order to represent them as rooted positional trees, we used the I.U.P.A.C. nomenclature rules (a set of rules was developed in [14]). An example of alkane representation is shown in Fig. 9. The vertexes in the trees have a maximum outdegree of 3, maximum indegree 1, and the maximum tree depth is 10. There is a total of 1331 vertexes in the data set.

The prediction of the boiling point yields to a regression task with a target associated to the root vertex of each tree. The target is the boiling point expressed in Celsius degrees ($^{\circ}C$) into the range $[-164, 174]$.

For the sake of comparison, we tested the prediction ability of the contextual versus the causal model using the same data set and learning parameters used for testing the causal model in [14]. Learning was stopped when the maximum absolute error for a single compound was below $8^{\circ}C$, or when a maximum number of hidden unit was reached (160 units for this set of experiments)⁷. All parameters have been chosen after an initial set of preliminary trials performed in order to determine an admissible range for the RCC models.

A 10-fold cross-validation on the dataset has been performed. For each fold 5 different learning trials were performed. In Table I the mean and the variance, over each fold and globally, of the mean absolute error obtained for the test set are reported. Specifically, the errors are expressed in $^{\circ}C$. Moreover, the average number of hidden units and the average maximum absolute test error, computed over all the folds, are reported as well.

In particular, it is possible to observe that the average of the mean test errors obtained by the contextual cascade correlation

⁷Actually, for few trials the maximum number of hidden units is reached before the maximum error on the training data set was below $8^{\circ}C$. However, we found that in such cases, both the mean error and the maximum error on the training data set are comparable to the values obtained with the trials that respect the stopping criterion on maximum train error.

TABLE I
CRCC vs RCC

	CRCC		RCC	
	Absolute Test Error		Absolute Test Error	
	Mean	Variance	Mean	Variance
fold-01	2.40	0.080	2.42	0.079
fold-02	1.85	0.200	3.52	1.210
fold-03	2.63	0.233	2.65	0.057
fold-04	2.44	0.173	2.93	0.150
fold-05	1.70	0.120	2.01	0.051
fold-06	2.26	0.116	3.16	0.179
fold-07	2.66	0.388	2.20	0.116
fold-08	2.95	0.026	2.47	0.102
fold-09	3.97	0.231	4.54	2.004
fold-10	2.78	1.768	2.75	0.163
Total Average	2.56	0.639	2.87	0.837
Average # of Hidden Units	137.8		140.1	
Average Max. Test Error	8.29		10.03	

is around 0.3°C less than the one obtained with the basic recursive cascade correlation, which corresponds to a relative improvement which is above 10%. Moreover, the CRCC is also more stable than the RCC since the variance on the obtained test mean error is lower than the RCC model. The improvement of CRCC in generalization is also testified by the relevant reduction of the average absolute maximum error which turns out to be very close to the stop criterion used for the training, i.e., 8°C . This does not hold for the RCC.

Moreover, notice that while improving the efficacy on the error results with the new model, the efficiency does not decrease, since the number of inserted hidden units by the two models, at the end of the training phase, is comparable. Of course, one hidden unit in CRCC will have more parameters than a hidden unit in RCC due to the contextual connections.

An example of comparison of learning curves between CRCC and RCC is shown in Fig. 10. In the figure the typical behavior of the two cascade models, reporting the mean training and test absolute errors obtained by the two models during learning as a function of the number of inserted free parameters, is shown. Here we have plotted the error curves versus the number of free parameters in order to have a fair comparison between the two models. In fact, when considering trees as input, the number of free parameters in the CRCC grows much faster than in RCC with the increase in the number of hidden units. The general formula for computing the number of free parameters of the models (with a single output unit) is

$$\# \text{ free param.} = (\text{out} + \text{in}) \frac{k(k-1)}{2} + (\text{out} + L + 2)k + 1,$$

where k is the number of inserted hidden units, and L is the size of the labels. For RCC, $\text{out} = \text{max_out_degree}$ and $\text{in} = 0$, while for CRCC, $\text{out} = \text{max_out_degree}$ and $\text{in} = \text{max_in_degree}$. In the alkanes dataset, $\text{max_out_degree} = 3$ and $\text{max_in_degree} = 1$.

These curves suggest that also the dynamical behavior of the CRCC is more effective than the dynamical behavior of the standard RCC. In fact, it can be noted that the CRCC model can obtain the same fitting or generalization results of

the RCC model, with much less parameters. Specifically, the generalization results obtained by the RCC at the end of the training phase, can be obtained by CRCC with a number of parameters which is a bit more than half.

C. Analysis of Internal Representations

In order to investigate the properties of the contextual encoding developed by the CRCC, we analyzed the internal representation, i.e. the output of the hidden units or state vector \mathbf{x} , obtained after training for the alkane dataset. Principal component analysis (PCA) was used to reduce the dimensionality of the representational space so to have the possibility to visualize the internal representations in a 2-D plot. Of course, in this way only global features of the internal representations are retained, however, we will see that this general features are enough to show how context is encoded by CRCC. Specifically, Fig. 11 shows the plot of the first two principal components of the internal representations generated from a sample experiment of the alkanes data set (on training set of fold-09). Similar plots, with differences restricted to rotations of the axes and scaling of the principal component values, can be obtained from different experiments with the same fold or using different folds. In the plot each point represents the code⁸ developed by CRCC for a (sub)structure of the alkanes data. i.e. the state vector $\mathbf{x}(v)$ for each vertex v in the data set. A (sub)structure can play either the role of chemical fragment or the role of compound (represented as a box in the plot). Some (sub)structures can be both compounds themselves and fragments belonging to more complex compounds, e.g. a single carbon group (C in Fig. 11) can represent both the methane and a group belonging to other compounds. Other (sub)structures only represent chemical fragments, and some of them may occur in the same compound, but in different positions (i.e., contexts), and/or they may occur into different compounds.

For the sake of clarity, in the plot we have highlighted only information relevant to some type of small and frequent substructures occurring in the data set. Specifically, we just consider structures with 1, 2, 3 and 4 vertexes. Different internal representations corresponding to the same structure (but occurring in different contexts) are grouped together by drawing them inside a solid line closed shape (cluster) associated with a picture showing the corresponding carbon tree. If a structure can play also the role of a chemical compound, the name of the compound is reported on top of the carbon tree. In this case, the internal representation associated to the compound, i.e., associated to the tree “without” context (in the sense that it is not part of a larger tree, as it happens when considering the same carbon tree as a fragment included in a larger chemical compound), is marked by a box. Moreover, different structures with the same number of nodes, are grouped by a dashed line closed shape.

Roughly, in Fig. 11, the simplest compounds, i.e., the ones with a small number of vertexes, are located on the left part of the plot, while more complex structures occur on the right

⁸To be precise, the first two principal components of the code developed by CRCC.

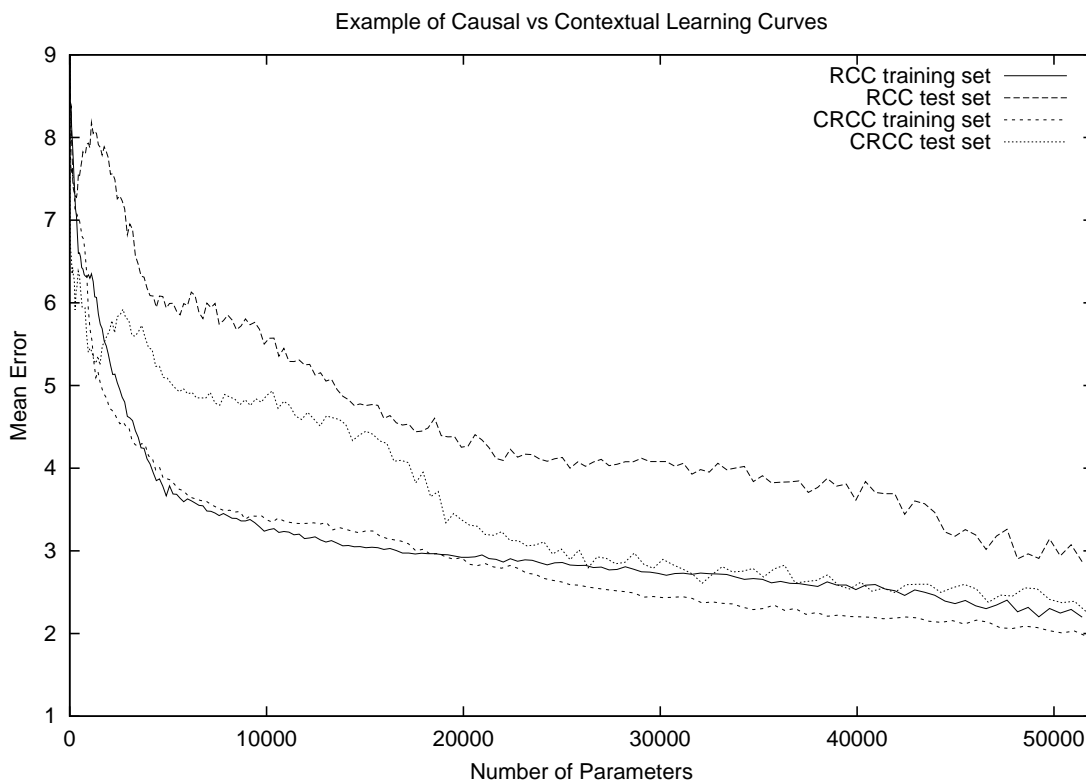


Fig. 10. Comparison of the learning curves obtained for RCC and CRCC models.

part of the plot. Therefore, most of the compounds are on the right part of the plot and most of the small fragments are on the left part of the plot. Since small fragments can belong to a large variety of compounds, i.e., contexts, our analysis will focus on the left part of the plot.

First of all, let us recall that RCC computes for each fragment, i.e. subtree, a unique code which is independent from the context where the fragment does occur. In fact, according to eq. 5 (and eq. 10), the generated state vector is not influenced by the location (inside a larger tree) where the considered subtree does occur. Hence, the analogous for RCC of each cluster shown in Fig. 11, necessarily will contain a single point that represents the state vector generated for the subtree.

Differently from RCC, and according to Propositions 1 and 2, each fragment in CRCC can be represented in different ways depending on the context, i.e., the position (inside a larger tree) where the subtree occurs. Thus, different points in the same cluster may represent either the same fragment occurring in different compounds, or different occurrences of the same fragment in the same compound.

In addition to that, the internal representations are spatially organized so to encode the complexity of the structure represented by each point, which in this prediction task is known to be correlated to the target information.

So, Fig. 11 shows that CRCC is actually able to develop internal representations that take into account the context in which each subtree does occur, and to organize these representations in a way which is functional to the solution

of the prediction task.

VI. CONCLUSION

We presented an extension of the RCC model based on the contextual analysis of structured domains, i.e., the Contextual Recursive Cascade Correlation (CRCC) model. The basic idea was to exploit contextual information already present in frozen hidden units. The proposed model constitutes the first example of a recursive neural model for processing of structured data able to exploit contextual information. In fact, both BRNN and the model defined in [8] can only deal with sequences, as well as BRCC, which by definition constitutes a special case of CRCC when applied to sequences.

We were able to formally elucidate how the “shape” of the contextual information evolves with the addition of hidden units, and to show that CRCC can compute contextual transductions which cannot be computed by RCC at all. In addition to that, we demonstrated that some causal supersource transductions, which cannot be computed by RCC, can be computed by CRCC, which on the other hand is able to compute all the transductions that can be computed by RCC.

Experimental results on controlled sequences have confirmed these results, and also have shown that CRCC is basically equivalent to RCC when considering a fully causal prediction task. Moreover, using the CRCC model we afforded a real-world task, i.e. the prediction of the boiling point of a set of alkanes, in order to show the improvements that can be obtained using a contextual approach versus a pure causal approach (standard RCC) when no information

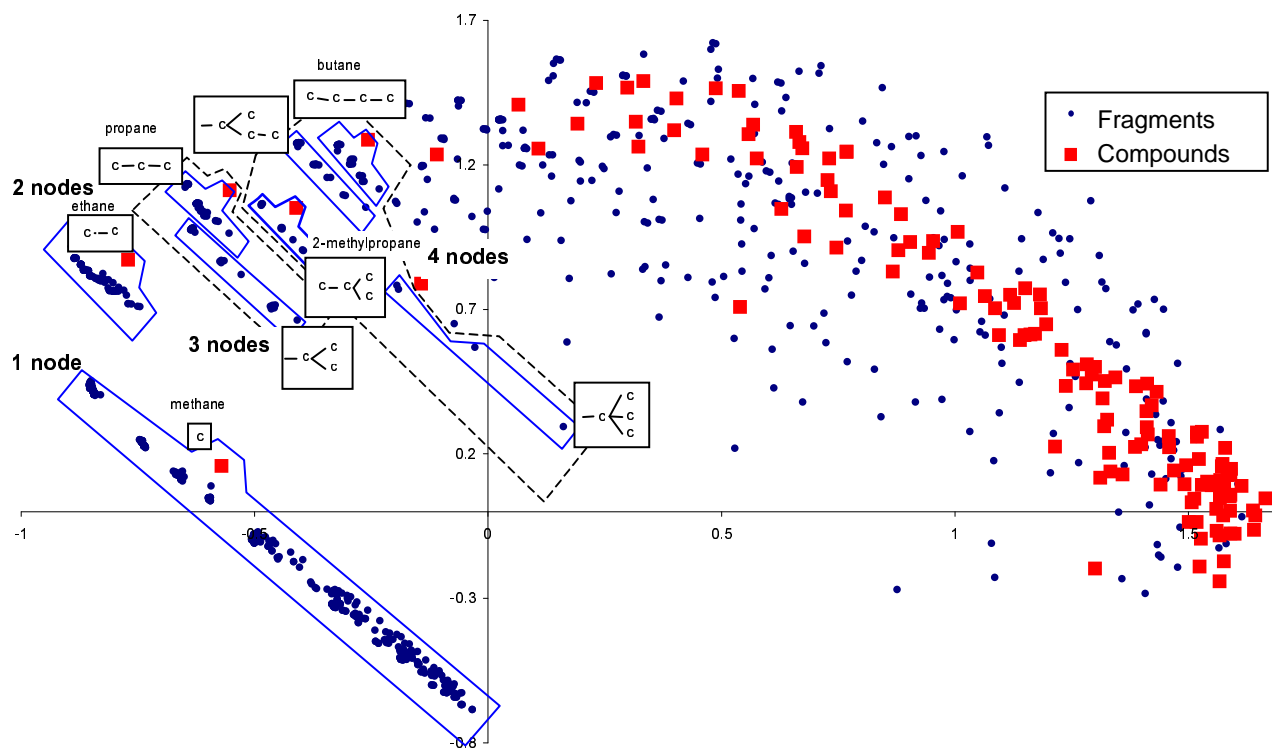


Fig. 11. PCA of internal representations of CRCC for Alkanes.

about the validity of the causality assumption is available. It should be noted that, even if the prediction task in this case was defined only on root vertexes, the CRCC model was able to develop internal representations taking into account the context. Specifically, the PCA analysis of the internal representations shows that each chemical fragment does get a different code in CRCC and the performance results allow us to conclude that these codes are more suited for the application task than the ones developed by RCC, which are independent from the context.

In addition to the possibility to exploit contextual information from a computational point of view, we think that the possibility to have information about the context in CRCC opens new ways to the error gradient flow through the structures, thus improving the efficacy of the gradient descent process.

In conclusion, both theoretical and experimental results suggest that the new model can be adopted as an alternative to the basic RCC model whenever it is not possible to guarantee the soundness of the causality assumption for the application domain under analysis. It should be stressed, however, that CRCC does not violate the causality constraint that allows to form an internal state of the dynamical system. In fact, the (causal) state information of frozen state variables is exploited in a contextual way from each new inserted state variable, and always according to the acyclic topology of the input graph. Moreover, the proposed model is asymmetric, since it uses a full causal flow of information from the descendants of a vertex, and only a partial flow of information from its ancestors. The partial flow of information is determined by a

partial view of the ancestors, being the size and shape of this view determined both by the number of inserted state variables and by the topology of the input graph. Interestingly, although the proposed asymmetric model can result in a reduction with respect to a fully (symmetric) contextual approach, it allows a dynamical control of the contextual information exploited by the model. As future work it would be interesting to devise contextual models that are symmetric in the way they generate new state variables. Concerning the computational power of the proposed model, it is well known that recurrent models based on Cascade Correlation cannot implement any finite state automata, while recurrent neural networks can (see [16] and [17]). This limitation, however, is a true limitation if no bound can be put on the size of the input data. In practical applications, however, the size of the input can be bounded, and so a Cascade Correlation based network can produce an automaton that is not minimal, but able to correctly work with inputs whose size is within the size bound. More work needs to be done concerning the function approximation capability of CRCC. For general recursive neural networks, general results on the approximation capability has been demonstrated in [18]. We are confident that some of those results can be extended to CRCC.

ACKNOWLEDGMENT

We thank Ah Chung Tsoi for very useful comments on a first version of this paper.

REFERENCES

- [1] A. Sperduti, D. Majidi, and A. Starita, "Extended cascade-correlation for syntactic and structural pattern recognition," in *Advances in Structural and Syntactical Pattern Recognition*, ser. Lecture notes in Computer Science, P. Perner, P. Wang, and A. Rosenfeld, Eds. Springer-Verlag, 1996, vol. 1121, pp. 90–99.
- [2] C. Goller and A. Küchler, "Learning task-dependent distributed structure-representations by backpropagation through structure," in *IEEE International Conference on Neural Networks*, 1996, pp. 347–352.
- [3] A. Sperduti and A. Starita, "Supervised neural networks for the classification of structures," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 714–735, 1997.
- [4] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing of data structures," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 768–786, 1998.
- [5] N. Qian and T. J. Sejnowski, "Predicting the secondary structure of globular proteins using neural network models," *Journal of Molecular Biology*, vol. 202, pp. 865–884, 1988.
- [6] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 3, pp. 328–339, 1989.
- [7] P. Baldi, S. Brunak, P. Frasconi, G. Pollastri, and G. Soda, "Exploiting the past and the future in protein secondary structure prediction," *Bioinformatics*, vol. 15, no. 11, pp. 937–946, 1999.
- [8] H. Wakuya and J. Zurada, "Bi-directional computing architectures for time series prediction," *Neural Network*, vol. 14, pp. 1307–1321, 2001.
- [9] A. Micheli, D. Sona, and A. Sperduti, "Bi-causal recurrent cascade correlation," in *Proc. of the Int. Joint Conf. on Neural Networks - IJCNN'2000*, vol. 3, 2000, pp. 3–8.
- [10] S. Fahlman, "The recurrent cascade-correlation architecture," in *Advances in Neural Information Processing Systems 3*, R. Lippmann, J. Moody, and D. Touretzky, Eds. San Mateo, CA: Morgan Kaufmann Publishers, 1991, pp. 190–196.
- [11] A. Micheli, D. Sona, and A. Sperduti, "Recursive cascade correlation for contextual processing of structured data," in *Proc. of the Int. Joint Conf. on Neural Networks - WCCI-IJCNN'2002*, vol. 1, 2002, pp. 268–273.
- [12] —, "A note on formal determination of context in contextual recursive cascade correlation networks," Dipartimento di Informatica, Università di Pisa, Italy, Tech. Rep. TR-04-04, 2004, URL: <ftp://ftp.di.unipi.it/pub/techreports/TR-04-04.ps.Z>.
- [13] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270–280, 1989.
- [14] A. Bianucci, A. Micheli, A. Sperduti, and A. Starita, "Application of cascade correlation networks for structures to chemistry," *Journal of Applied Intelligence (Kluwer Academic Publishers)*, vol. 12, pp. 117–146, 2000.
- [15] D. Cherqaoui and D. Villemin, "Use of neural network to determine the boiling point of alkanes," *J. Chem. Soc. Faraday Trans.*, vol. 90, no. 1, pp. 97–102, 1994.
- [16] C. L. Giles, D. Chen, G.-Z. Sun, H.-H. Chen, Y.-C. Lee, and M. W. Goudreau, "Constructive learning of recurrent neural networks: Limitations of recurrent cascade correlation and a simple solution," *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 829–836, July 1995.
- [17] A. Kuchler, "Adaptive processing of structural data: From sequences to trees and beyond," Ph.D. dissertation, Faculty of Computer Science, University of Ulm, Germany, 1999.
- [18] B. Hammer, *Learning with Recurrent Neural Networks*, ser. Lecture Notes in Control and Information Sciences, Vol. 254. Springer-Verlag, 2000.



Alessio Micheli received the laurea degree and the Ph.D. in Computer Science from the University of Pisa in 1998 and 2003, respectively. He is currently Research Fellow at the Computer Science Department of the University of Pisa. He has been visiting fellow at Wollongong University (AUS) and at the NCRG of Aston University (UK). His research interests include Machine Learning, Neural Networks, Structured Domains Learning, Recursive Neural Networks, Kernel-based learning for non-vectorial data, applications to Natural Science and Biochemistry, QSPR/QSAR (Quantitative Structure Property/Activity Relationships) Analysis, Cheminformatics and Drug Design. He is involved in several National and European projects in the field of Computational Intelligence and Cheminformatics. He is a member of the IEEE Neural Networks Society and of the QSAR and Modelling Society.



Diego Sona received the Laurea degree in computer science from the University of Pisa, Italy, in 1996, and the Ph.D. degree from the University of Pisa in 2002. He is currently a research scientist at the Automated Reasoning Systems division of the Center for Scientific and Technological Research (ITC-irst), Trento, Italy. His research interests include neural networks, pattern recognition, information retrieval, and relational learning.



Alessandro Sperduti received his education from the University of Pisa, Italy ("laurea" and Doctoral degrees in 1988 and 1993, respectively, all in Computer Science.) In 1993 he spent a period at the International Computer Science Institute, Berkeley, supported by a postdoctoral fellowship. In 1994 he moved back to the Computer Science Department, University of Pisa, where he was Assistant Professor, and Associate Professor. Currently, he is Full Professor at the Department of Pure and Applied Mathematics, University of Padova. His research

interests include pattern recognition, image processing, neural networks, kernel methods, and hybrid systems. In the field of hybrid systems his work has focused on the integration of symbolic and connectionist systems. He contributed to the organization of several workshops on this subject and he served also in the program committee of several conferences on Neural Networks, Machine Learning, Artificial Intelligence, and Information Retrieval. Alessandro Sperduti is the author of around 90 refereed papers mainly in the areas of Neural Networks, Fuzzy Systems, Pattern Recognition, and Image Processing. Moreover, he gave several tutorials within international schools and conferences, such as IJCAI '97, IJCAI '99, IJCAI '01. He acted as Guest Co-Editor of journal special issues on Connectionist Models for Learning in Structured Domains. He is a member of the Executive Board of the Italian Neural Network Society (SIREN) and the European Neural Networks Society.