Designing a Neural Network Primitive for Conditional Structural Transformations

Alexander Demidovskij and Eduard Babkin(✉)

Higher School of Economics, Bolshaya Pecherskaya Street 25/12,
Nizhny Novgorod 603005, Russia
{ademidovskij,eababkin}@hse.ru

Abstract. Among the problems of neural network design the challenge of explicit representing conditional structural manipulations on a sub-symbolic level plays a critical role. In response to that challenge the article proposes a computationally adequate method for design of a neural network capable of performing an important group of symbolic operations on a sub-symbolic level without initial learning: extraction of elements of a given structure, conditional branching and construction of a new structure. The neural network primitive infers on distributed representations of symbolic structures and represents a proof of concept for the viability of implementation of symbolic rules in a neural pipeline for various tasks like language analysis or aggregation of linguistic assessments during the decision making process. The proposed method was practically implemented and evaluated within the Keras framework. The network designed was tested for a particular case of transforming active-passive sentences represented in parsed grammatical structures.

Keywords: Tensor product representations · Artificial neural networks · Linguistic decision making · Natural language processing

1 Introduction

Despite existing advances in mathematical models and technologies of deep learning, neural-inspired computational architectures researchers outline a considerable gap between connectionist sub-symbolic and logic-based symbolic approaches to representation and higher level reasoning [2,3,15]. From the symbolic perspective, it seems quite apparent that human cognition operates with complex symbolic data structures: graphs, trees, shapes, grammars etc., performs symbolic manipulations with means of symbolic logic. However the processing of these structures in mind is performed on the neural level. At the same time, existing attempts to build artificial neural systems lose in terms of representational compositionality [14]. In the framework of neural-symbolic computation, there is the proved *in principle* equivalence between dynamical systems with distributed representations and symbolic systems in terms of representational or problem-solving capabilities [19]. At the same time there are no exact rules of

S. O. Kuznetsov et al. (Eds.): RCAI 2020, LNAI 12412, pp. 117–133, 2020.
https://doi.org/10.1007/978-3-030-59535-7_9

obtaining a sub-symbolic counterpart to symbolic models and vice versa. In practice, little attention was given to development of practically achievable software implementations of such sub-symbolic models.

In the neural-symbolic paradigm Artificial Neural Networks (ANN) are the means of parallel distributed computation and robust learning. In this field there is an important scientific task of constructing neural networks that perform significant intellectual tasks without preliminary training stage [25] in massively parallel computation environments like multi-agent systems or Internet Of Things (IoT). Each component of such systems plays a role of a single neuron or a small sub-network [33]. These distributed computational platforms should be not only distributed, but also robust to unit failures, self-improving in time and avoid central control. The first step towards solving that task would be design of ANN capable of producing an exact solution for a selected motivating problem, which combines different intellectual operations on complex symbolic structures.

The task of multi-criteria linguistic based decision making can be selected as an example of appropriate motivating problem [10,32]. Creation of monolithic neural-symbolic systems for various expert and decision support systems is an actual task [16,25]. Linguistic assessments aggregation is a key element of fuzzy decision making models [5] and it seems to be a hard requirement for any neural-symbolic decision support system.

In order to put step forward on the way of obtaining a practical sub-symbolic solution a bottom-up approach is proposed. According to that approach separate neural networks which perform critical sub-tasks of manipulation without training (called *primitives*) should be designed and combined within a single meta-network capable of producing a final solution of the multi-criteria choice problem. The term 'network' is widely used in different contexts, so we use attribution 'neural' in our own method to clearly specify that our approach is based on combination of existing modules that implement the functionality of neurons. The application of existing implementations of such modules enables actual software design of our proposals. Following the approach proposed a certain schema of combination of required primitives was designed, see Fig. 1.

There are several recent advances in the field that allow building such sub-symbolic solutions and prove viability of the proposed approach. Usually the integrated symbolic and sub-symbolic flow consists of following steps:

1. encoding the symbolic structure as a distributed representation with a neural network. In [8] a new encoder design was proposed for the simple structure that has only one nesting level.
2. flattening the distributed representation to a vector format with a neural network [12].
3. performing domain specific analysis of the structure on the neural level. For example, aggregation of linguistic assessments during decision making or voice identification of a sentence during linguistic analysis.
4. structural manipulations on the neural level, for example, joining of two trees in one [12].
5. decoding the new structure from a distributed representation to symbolic level [8].

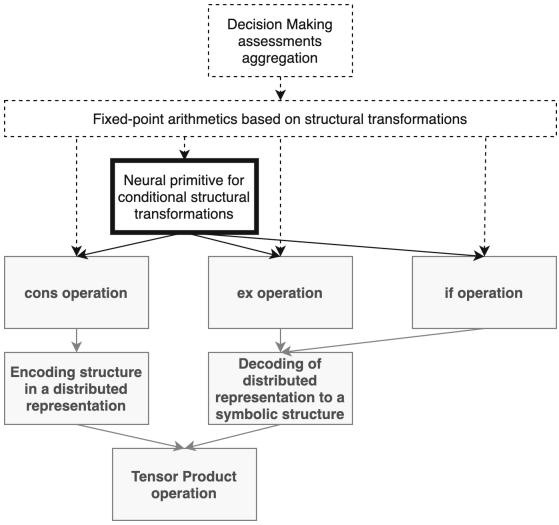


Fig. 1. Evolution of sub-symbolic methods required for expression of linguistic assessments aggregation during decision making process. Grayed cells refer to the existing research, thick-bordered cell represents current research direction, dashed-bordered cells stand for directions of further research.

Aforementioned papers use the idea of compiling neural networks, therefore, training step is not required [23]. Such compiled networks produce results equivalent to those produced by the symbolic algorithm, while other, and more popular nowadays, examples of trainable neural-symbolic systems perform reasoning probabilistically [22].

This paper offers a novel design of a neural network which is capable of performing conditional transformations of arbitrary structures expressed in terms of Tensor Product Variable Binding. Such neural network can be embedded to more complex networks for different computational tasks, thus we call it a neural network primitive. In particular, the proposed neural network primitive consists of a cascade of three small neural networks: the first one extracts the marker from an input structure for a conditional module, the second network performs conditional extraction of specific structural elements and the final model constructs a new structure from elements of the input one. We present design and evaluation results of the network capable of logical branching. This primitive is based on the analysis of arbitrary symbolic structures and can be considered as a sub-symbolic equivalent of IF logical operator in traditional programming languages. Careful engineering of the network provides better comprehensibility and maintenance, as well as potential reuse for solving other symbolic operations translation to the sub-symbolic level. The network inference result is a symbolic structure encoded in a form of distributed representation. This method of generating neural networks that are capable of encoding and manipulating structures

on the tensor level can be used in broad range of cognitive systems. The designed network addresses an applied task of detecting voice of English language.

The article is structured as follows. Section 2 presents the context of the research and outlines the most relevant achievements in the field of encoding symbolic structures in distributed representations. In Sect. 3 the proposed design of a neural network primitive is presented. Section 4 contains overview and analysis of experimental results with the elaborated sub-symbolic model. Conclusions and further directions of research are given in the final part of the paper.

2 Distributed Representations of Symbolic Structures

There are multiple ways of transforming a generic recursive structure in a distributed format. Foundations were developed in Tensor Product Variable Binding (TPVB) [27]. Later TPVB approach became an inspiration for such representation methods as Holographic Reduced Representations (HRRs), Binary Spater Codes [4] and Vector Symbolic Architectures (VSA) [15]. These ideas were generalized as Integrated Connectionist/Symbolic cognitive architecture [29] and later resulted in Gradient Symbolic Computation (GSC) framework [6,28]. Huge investments are made in research of Knowledge Base (KB) translation to a form of First-Order Logics (FOLs) with strictly defined rules and further encoding of such expressions [26,31]. FOL expressions can be represented as a labelled directed graph which may be translated to distributed representation with one of existing conjunctive non-temporal binding mechanism proposed in [24].

We consider TPVB as a relevant approach in our research due to its generic nature and continuing extensions of the original ideas. For example, recent works demonstrate applicability of TPVB for such tasks as image captioning [17] or question answering [21]. Also, TPVB can be used as the mechanism of structure recovery from a distributed representation to analyse the neural network ability to generalize and include structural properties of objects embeddings for which it learns to create [30]. Recent advances in deep learning architectures allowed several research groups to investigate an idea of learning structural embeddings so that a neural network decides how to encode the structure [20].

From the encoding strategy perspective, TPVB is a set of predefined rules for constructing a distributed representation of an arbitrary symbolic structure with no information loss and corresponding symbolic operations in the weights of the network so that the resulting structure emerges as a network output. Tensor Product Variable binding allows to build recursively such distributed representations on top of atomic elements: fillers and roles.

Definition 1. *Fillers and roles [27]. Let S be a set of symbolic structures. A role decomposition F/R for S is a pair of sets (F, R), the sets of fillers and roles, their Cartesian product $F \times R$ respectively and a mapping:*

$$\mu_{F/R} : F \times R \mapsto Pred(S); (f, r) \mapsto f/r. \tag{1}$$

For any pair $f \in F, r \in R$, the predicate on S $\mu_{F/R}(f, r) = f/r$ is expressed: f fills role r.

Definition 2. *Let s be a symbolic structure that consists of pairs $\{f_i, r_i\}$, where f_i represent a filler and r_i represents a role. Tensor product ψ is calculated in the following way:*

$$\psi = \sum_i f_i \bigotimes r_i \tag{2}$$

Given notation of roles and fillers, there are two primitive operations defined: *cons* and *ex*. The $cons(p, q)$ operation takes two trees as arguments and creates another tree that has tree p as a left child, or in terms of TPRs gets the role r_0, and q as a right child, or in terms of TPRs gets the role r_1. The important requirement is to select roles vectors so that they are linearly independent. The same requirement applies for the set of fillers vectors. As it was proved in [29] the *cons* operation can be expressed as a matrix-vector multiplication.

Definition 3. *Let r_0, r_1 denote role vectors, p, q - symbolic structures. Then, joining operation cons is defined:*

$$\begin{aligned} cons(p, q) &= p \bigotimes r_0 + q \bigotimes r_1 \\ &= W_{cons_0} p + W_{cons_1} q \end{aligned} \tag{3}$$

Definition 4. *Let r_0 denote a role vector, A is a length of any filler vector. Then joining matrix W_{cons_0} is calculated in the following way:*

$$W_{cons_0} = I \bigotimes 1_A \bigotimes r_0, \tag{4}$$

where I is the identity matrix on the total role vector space, 1_A is the identity matrix $A \times A$. W_{cons_1} matrices are defined in the manner similar to the W_{cons_0} matrices that join two sub-trees in one structure [29]. Extraction operation *ex* is defined in analogous way, however, it is used to extract an element stored in the tree by the given role, for example $ex_0(p)$ extracts the child of tree p that is placed under role r_0. The only difference in formulation of W_{ex_0} matrix is that dual role vectors are used instead of direct roles: r_0 or r_1. This operation is used in both extraction branches that are described in Sect. 3.

Definition 5. *Let r_0 denote a role vector, $s = cons(p, q)$ - a symbolic structure. Then, extraction operation ex_0 is defined:*

$$ex_0(s) = ex_0(cons(p, q)) = p, \tag{5}$$

Definition 6. *Let u_0 denote an extraction (or so-called unbinding [29]) vector, which belongs to the basis dual to the basis which includes r_0, A is a length of any filler vector. Then extraction matrix W_{ex_0} is calculated in the following way:*

$$W_{ex_0} = I \bigotimes 1_A \bigotimes u_0, \tag{6}$$

Aforementioned operations *cons*, ex_0, ex_1 are equivalent to operations over lists in software general-purpose functional programming languages like Lisp:

cons, *car*, *cdr*. These are universal operators that allow implementation of huge set of algorithms. Considering support of conditional operator, the representational power of these operators is even bigger. Therefore implementation of these operations, or equivalent *cons*, ex_0, ex_1, on the neural level opens the horizon for neural-symbolic computation of symbolic algorithms built on top of *cons*, *car* and *cdr*. Indeed, the matrices W_{ex_0} and W_{cons_0} determine the weights of a neural network, capable performing corresponding symbolic operations. In [29] a formal specification of their combination is given to produce a neural network equivalent for an arbitrary list manipulation operation.

Many models are already known for representation of linguistic intelligence capabilities in terms of sub-symbolic neural computations, like [4,6,28] or pre-trained deep bidirectional representations from unlabeled text like [13]. Among different approaches we distinguish Active-Passive Network (APNet) proposed by P. Smolensky [18]. This network performs two tasks: classification of a voice of a sentence and semantic analysis that allows to extract nominal subject and direct object dependencies in a form of a structure. Further analysis of this architecture through lenses of neural-symbolic computation identifies that the network relies on existence of three actions: extraction of specific elements of a structure, conditional branching, and construction of new structure from elements of input structures. Therefore, elaboration of a software design for this theoretical architecture allows further development of the field and construction of neural-symbolic means of performing linguistic assessments aggregation.

This network was designed to work over semantic parse trees, estimate the voice of a sentence and construct a predicate-calculus expression that contains information about verb V and relationship between agent A and patient P parts of such trees (Fig. 3a, Fig. 3b). It was proposed to use existence of Aux filler at a role r_{001} as a universal marker of Passive voice (Fig. 3a). Such a notation of roles should be read as left child (0) of the left child (0) of the right child (1) of the root of the given structure. Aforementioned primitive ex can be used to extract passive voice marker from a given sentence (7).

$$PassiveMarkerF(s) = ex_0(ex_0(ex_1(s))) \tag{7}$$

Once the voice of the input sentence is defined, it is possible to construct a desired predicate-calculus expression. In order to do that, each filler that is a part of such target structure (Fig. 3c) should be extracted from the corresponding input structure. For the Passive voice case extraction rules (8) for each fillers are different compared to extraction rules of analogous fillers in Active voice case (9).

$$\begin{aligned} V = V_{passive} &= ex_1(ex_0(ex_1(s))) \\ A = A_{passive} &= ex_1(ex_1(ex_1(s))) \\ P = P_{passive} &= ex_0(s) \end{aligned} \tag{8}$$

$$\begin{aligned} V = V_{active} &= ex_0(ex_1(s)) \\ A = A_{active} &= ex_0(s) \\ P = P_{active} &= ex_1(ex_1(s)) \end{aligned} \tag{9}$$

Finally, once the fillers are found, the *cons* operation is used to obtain the desired structure (Fig. 3c).

$$AP(s) = cons(V, cons(A, P)) \tag{10}$$

The aim of this research is elaboration of the neural network primitive capable of preforming several symbolic operations: conditional branching and extraction of elements. The aim of Active-Passive Network is intellectual analysis of parsed grammatical structures that implies application of symbolic operations mentioned above. Therefore, the design of the neural primitive proposed in this paper reflects the applied task of Active-Passive Net while persisting generality of the solution for future re-use in building neural-symbolic decision support system. Proposed neural design is covered in the next section.

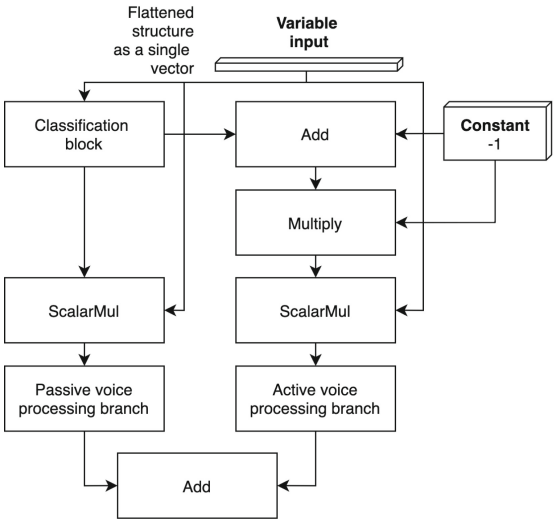
3 Proposed Neural Design of Network with Conditional Branching

3.1 Network Architecture

Current research is targeted to propose a novel design of neural network primitive that is capable of performing various tasks on a distributed representation of a symbolic structure: extraction of elements, condition branching and construction of a new structure. As a demonstration the Active-Passive Network was chosen. The proposed design is shown on Fig. 2a. In general the neural network consists of three important blocks: one classification and two processing branches. The classification branch is aimed at identification of whether the given sentence is in Passive or Active voice. At the same time, processing branches extract necessary elements of a symbolic structure from its distributed representation in order to construct a predicate-calculus expression. Each branch is considered separately below.

Classification Branch. According to the semantic parse trees that are obtained from raw sentences there is an obvious marker that can be used for identification of a voice in a sentence. As it was mentioned in [18], existence of *Aux* is a clear marker of a sentence in a Passive voice iff it is placed as a left-child-of-left-child-of-right-child-of-root. When TPRs notation is applied to such structures, marker is expressed as existence of *filler Aux* with a positional role r_{001}. The overall idea of the classification branch of the proposed Active-Passive network is to check this filler on the given position and output a binary value, where *1* represents that given sentence is in Passive voice and *0* means that it is in Active voice. The structure of this branch is shown on Fig. 2b.

Inputs. Classification branch has one variable input that is an one-order tensor representing the encoded symbolic structure of the semantic parse tree. In our approach construction of such parse tree on the basis of an initial text is



a. Overall design of a network. b. Classification block design.

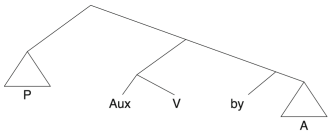
Fig. 2. Proposed architecture of a neural network primitive the performs: extraction of symbolic counterparts, conditional branching and construction of a new structure from input structure elements.

considered as an external task. The semantic parse tree can have an arbitrary structure, and algorithms from [12] can be used to encode it to the vector format. Apart from one variable input the network has three constant inputs that contain weights for the extraction operation. Those three inputs are by design matrices for operation ex: W_{ex_0} and W_{ex_1}. Matrices are generated in advance according to the recursive definition (4).

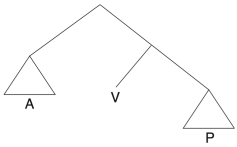
Shift Block. The classification branch of the Active/Passive Network contains three shift blocks. Each of them receives current tree representation and performs extraction of the particular child of this tree. More specifically, as the aim of the branch is to find the *Aux* filler, there is a sequence of three extraction or 'shift' blocks that perform retrieval of left child, left child and right child correspondingly. The output of each shift block is the distributed representation of the particular part of the input structure.

Outputs. As it was mentioned above the classification branch can be considered as a self-contained neural network capable of performing a basic task of classifying the voice of a sentence. The only output of a classification branch is binary and equals *1* when sentence is in Passive voice. From the engineering perspective, output is implemented as a Keras Mask layer[1].

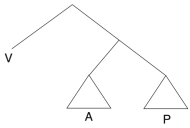
[1] From now on network description contains terminology accepted in the Keras [7] and TensorFlow [1] software frameworks.



a. Input sentence in the Passive voice.



b. Input sentence in the Active voice.



c. Output structure produced by a network.

Fig. 3. Types of symbolic structures used as inputs/output in a proposed neural network primitive.

Processing Branch. Processing branches play an important role in the Active-Passive Network and they perform construction of a new structure. It is impossible to exaggerate the importance of the fact that two most valuable structural operations, such as joining of two trees in a bigger one and extraction of the tree elements, can be performed on a sub-symbolic level. These two operations are key ones as a majority of other operations can be expressed with a help of just *cons* and *ex* primitives. There are two parallel processing branches due to the fact that sentences in Passive and Active voice have different architecture that is reflected in Fig. 3a and Fig. 3b respectively. As a result of each processing branch it is required to obtain a predicate-calculus expression that itself is a structure and consists of three elements: agent, patient and verb. Refer to Fig. 3c as a visual representation of the network output.

Inputs. Each of these branches accepts the one-order tensor that is used for further manipulations and construction of a new structure. This vector either contains the distributed representation of the input structure or is a placeholder filled with zeros. This completely depends on the type of the branch and results of the model classification.

When the sentence is in Passive voice then the input of the Passive voice processing branch is exactly the distributed representation of the input structure and Active voice processing branch receives a placeholder. In case a sentence is in Active voice, the first processing branch accepts a placeholder with zeros while the second branch receives the embedding of the structure. This behaviour is handled by the masking head before the Active voice processing branch. The idea behind such a switch implemented on the neural level is that in Active voice the sentence has a completely different structure and in order to extract particular filler, for example V ($Verb$), the completely different set and sequence of operations is needed.

The overall architecture of the processing branch is reflected in Fig. 4. In general, it consists of two parts: extraction and joining logic. Extraction part is

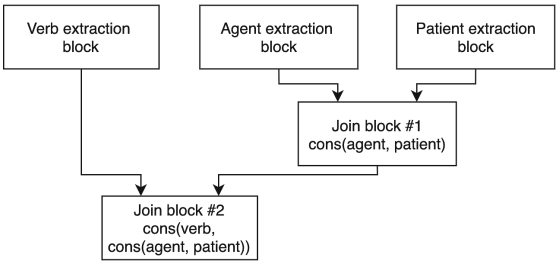


Fig. 4. Architecture of processing branch.

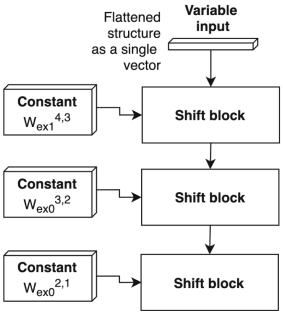
voice-specific and is different between sentences in Passive and Active voice. At the same time construction part is common and does not depend on the type of input sentence. Details about each type of processing branch elements are covered below.

Extraction Blocks. As it was stated above, extraction rules are fully defined by the type of the sentence. For example, in order to extract verb from the sentence in Passive voice it is required to perform operation $ex_1(ex_0(ex_1(s)))$ while for the Active voice sentence it is enough to execute $ex_1(ex_0(s))$ in order to extract V ($Verb$). Difference in operations results in different architecture of extraction branches. Figure 5a reflects extraction of V in Passive voice sentence and Fig. 5b shows extraction of the same filler in an Active voice sentence.

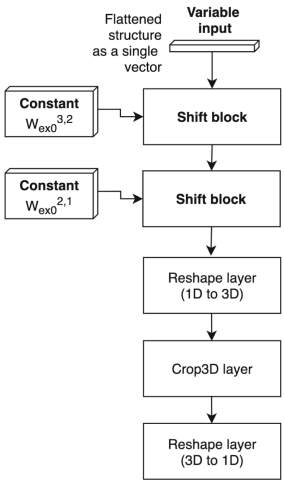
It is important to note that the idea of the extraction block is to extract a given filler from the input structure encoded in a form of a vector. This brings several limitations. The biggest one is that input sentences have to be encoded in a vector of the same size. This implies that Active voice sentence have to be encoded with an additional placeholder tensor in order to be on par with the Passive voice sentence. As a consequence, extraction of the left child of the root results not in the vector representing a filler A ($Agent$) but in a vector of bigger format. In order to satisfy the condition that any extraction branch results in the vector representing a filler, the additional cropping operation has to be introduced. In this task, it is specific for Active voice branch and is reflected on the Fig. 5b.

There are three extraction blocks in each extraction branch in the proposed design on the Active-Passive network due to the fact that the resulting structure (Fig. 3c) is constructed from three fillers.

Join Blocks. The next step after all the required fillers are extracted is to perform construction of the new structure that stands for the predicate-calculus expression (Fig. 3c). It is suggested to reuse existing mechanism of joining trees in one bigger tree with appropriate role assignment [8,12]. A scheme of such a join block is shown in Fig. 6. Each joining block accepts two variable inputs each representing the tree that would become a part of a resulting structure. Also, there are two constant inputs that represent the joining matrix (4) and



a. Case of sentence in a passive voice



b. Case of sentence in an active voice

Fig. 5. Example architecture of extraction branches

one is used as a utility constant when there is mismatch between the two variable inputs. A final operation of this branch is an addition operation that joins shifted sub-trees in a single distributed representation. The block shown in Fig. 6 represents the second join block in Fig. 4.

Outputs. The output of each processing branch is either a distributed representation of a new structure or a placeholder of the same dimension. According to the aforementioned condition, distributed representation of a new structure is presented as an output of the Passive branch if a sentence is in Passive voice and, in contrary, in case a sentence is in Active voice, distributed representation of a new structure appears as an output of an Active voice processing branch. The final sum of outputs of both branches is needed because there is no a-priori knowledge about input sentence voice. The output of the Active-Passive neural network is a distributed representation of a valid structure and it can be decoded with an existing methods of extracting fillers [18,27].

3.2 Network Analysis

The proposed architecture demonstrates practical feasibility of expressing symbolic operations in a distributed and robust manner that is a neural network. It is implemented with the Keras framework [7] and Tensorflow backend [1]. There is a clear separation of responsibilities in the network and selected parts can be considered as standalone neural networks that solve more specific task. Moreover, the proposed neural network combines both joining and extraction operations and also contains conditional branching for handling conditions on

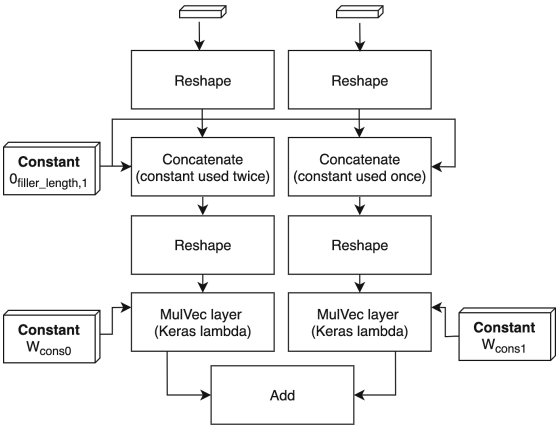


Fig. 6. Join block architecture

the sub-symbolic level. However, there are several important aspects that should be considered as directions of further research:

1. common sub-graphs sharing. More strict analysis of the proposed architecture would reveal that parallel processing branches as well as classification branch contain execution sub-graphs that are common by the operation they perform. At the same time, in the neural network this property is not re-used, although more smart construction rules would allow to decrease computational complexity of the solution. At the same time, there should be a certain trade-off because once those sub-graphs become common across all the branches the overall network cannot be expressed as a cascade of smaller neural networks that can be also executed separately on different instances of hardware. The maximum performance gain and various optimization options should be investigated in order to understand the best possible trade off between network architecture expressiveness and utility.
2. generation of the network that solves the task described by arbitrary combination of primitive operations like *cons* or *ex*. The overall goal of research in this field is building a bridge between symbolic and sub-symbolic computations. Therefore, elaboration of a flexible network generator would allow to make a step forward in this direction. However, this is a challenging task that requires thorough research and identification of minimum set of operations that should be supported by such generator in order to express arbitrary symbolic algorithm in a distributed manner.
3. comparison of semantic tree embeddings obtained with TPRs approaches and deep learning. Due to the fact that TPRs allow getting embeddings of a sentence it would be extremely interesting to analyse validity of using them in a generic text analysis task. However, in order to prove it, those embeddings have to be really representative and reflect the context of the usage.

This direction is the most open one and goes far beyond elaboration of neural architectures for expressing symbolic operations.

The proposed neural design considers generation of the network that is not trainable and produces results equivalent to those produced by the symbolic algorithm, compared to other examples of trainable neural-symbolic systems [22] that perform reasoning probabilistically. Network generation, or compilation, is not a novel topic in the field of neural architectures applied to neural-symbolic computation, for example networks compiled from logic formulas [23]. Such networks are characterised by thousands of neural units and connections between them. However, due to the fact that neural networks are by design distributed and parallel, it is a topic of another research to analyze large-scale gains of massive parallelism that would make such architectures computationally justified. The Active-Passive Network works on top of distributed Tensor Product Representations that provide compact and scalable binding capabilities for expressing symbolic structures and operations on them.

4 Evaluation of the Proposed Method

In order to explain application of the developed primitive for conditional structural transformations, we embedded it inside the APNet for active-passive voice recognition. In [9] detailed examples of actual text sentences are given those parsing trees correspond to symbolic structures from Fig. 3a and Fig. 3b. In order to encode each structure in a distributed form each filler and role are defined as linearly independent vectors, exact values are selected randomly as they do not play any role other than encoding a particular element.

$$\begin{aligned} A &= [7\,0\,0\,0\,0]\; V = [0\,4\,0\,0\,0] \\ P &= [0\,0\,2\,0\,0]\; Aux = [0\,0\,0\,5\,0] \\ by &= [0\,0\,0\,0\,3]\; r_0 = [10\,0]\; r_1 = [0\,5] \end{aligned} \tag{11}$$

According to the rules defined in [27] and using the lightweight binding network proposed in [8,12], both sentences are translated to the distributed representation (12), (13).

$$\begin{aligned} S_{Passive} = & \\ & \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} [0,0] \\ [0,0] \\ [20,0] \\ [0,0] \\ [0,0] \end{bmatrix}, \begin{bmatrix} [[0,0],[0,0]] \\ [[0,0],[0,0]] \\ [[0,0],[0,0]] \\ [[0,0],[0,0]] \\ [[0,0],[0,0]] \end{bmatrix}, \\ & \begin{bmatrix} [[[0,0],[0,0]],[[0,0],[0,875]]] \\ [[[0,0],[0,0]],[[0,1000],[0,0]]] \\ [[[0,0],[0,0]],[[0,0],[0,0]]] \\ [[[0,2500],[0,0]],[[0,0],[0,0]]] \\ [[[0,0],[0,750]],[[0,0],[0,0]]] \end{bmatrix} \end{aligned} \tag{12}$$

$$S_{Active} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} [70, 0] \\ [0, 0] \\ [0, 0] \\ [0, 0] \\ [0, 0] \end{bmatrix}, \begin{bmatrix} [[0, 0], [0, 0]] \\ [[0, 200], [0, 0]] \\ [[0, 0], [0, 50]] \\ [[0, 0], [0, 0]] \\ [[0, 0], [0, 0]] \end{bmatrix} \tag{13}$$

One of the requirements for correct inference of the classification branch is that representation of active and passive sentences should be of the same size. For that, the Active sentence representation is extended with an additional tensor filled with zeros. When classification branch is executed, the *Aux* filler is extracted. For the Passive voice sentence this filler is extracted without any loss, while for the Active voice sentence it is absent and application of the extraction rules results in the vector of the same size as any of fillers is but filled with all zeros. As a result, the branch outputs *1* for the first sentence and *0* for the second one.

Another important aspect is the output of the model. As it was described above, the output of the model is the distributed representation of the new structure that stands for the predicate-calculus expression. For this example, such representation would be the same for both sentences (14) as we encode fillers but not particular words of those sentences.

$$S_{Result} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} [0, 0] \\ [0, 40] \\ [0, 0] \\ [0, 0] \\ [0, 0] \end{bmatrix}, \begin{bmatrix} [[0, 350], [0, 0]] \\ [[0, 0], [0, 50]] \\ [[0, 0], [0, 00]] \\ [[0, 0], [0, 0]] \\ [[0, 0], [0, 0]] \end{bmatrix} \tag{14}$$

Finally, each filler can be extracted according to the rules defined in [27]. The proposed neural network solves the task of defining the voice of the sentence as well as allows constructing the new structure from the elements of input sentence in a scalable and robust manner. From the performance perspective, grammatical parse trees encoding takes approximately 1.3 ms, Active-Passive Network generation and inference take 5.5 and 1.4 ms respectively. Benchmarking was made with the following setup: Intel(R) Core(TM) i7-4770HQ CPU 2.20GHz (not fixed frequency). Implementation of the neural network primitive is available as an open-source project[2]. Experimentation results show that the developed neural network primitive enables conditional distributed computations as required.

The method proposed imposes limitations to the maximum depth of the processed symbolic structures. Thus a designer should know the maximum depth of the structures in advance and use it as a parameter for generating the corresponding primitive.

[2] https://github.com/demid5111/ldss-tensor-structures.

5 Conclusion

The results obtained in this paper contribute to the neural-symbolic paradigm by demonstrating execution of symbolic operations on the sub-symbolic level [2]: translation of symbolic knowledge into the network, executing the network or performing reasoning and knowledge extraction from the network output. Our method enables automatic design of Keras-based software implementation of a distributed computational structure for a generic primitive for conditional structural transformations. Authors consider that primitive as an important building element of a linguistic-based decision-making support system as it was depicted in Fig. 1. A recent work [11] demonstrates joint application of the developed primitive with other elements of that scheme for distributed implementation of arithmetic operations, which in own turn will be applied for distributed implementation of aggregation operators. The results of these works contribute to the support of the principal hypothesis, which states that linguistic information aggregation can be expressed in the form of structural manipulations and if it is true, then this aggregation step of multiple decision-making methods can be expressed in a distributed and robust manner of sub-symbolic, or connectionist, computation. The development of such models would allow for construction of fully-integrated monolithic neural-symbolic systems. At the same time, there is an important practical aspect of the selected binding mechanism (TPRs) that should be further analyzed: the size of the distributed representations and the dependence of this size on the size of the input structures that should be encoded.

Acknowledgements. Authors sincerely appreciate all valuable comments and suggestions given by the reviewers. The reported study was funded by RFBR, project number 19-37-90058.

References

1. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). http://tensorflow.org/. Software available from tensorflow.org
2. Besold, T.R., et al.: Neural-symbolic learning and reasoning: a survey and interpretation. arXiv preprint arXiv:1711.03902 (2017)
3. Besold, T.R., Kühnberger, K.U.: Towards integrated neural-symbolic systems for human-level AI: two research programs helping to bridge the gaps. Biol. Inspired Cogn. Archit. **14**, 97–110 (2015)
4. Browne, A., Sun, R.: Connectionist inference models. Neural Netw. **14**(10), 1331–1355 (2001)
5. Cheng, P., Zhou, B., Chen, Z., Tan, J.: The topsis method for decision making with 2-tuple linguistic intuitionistic fuzzy sets. In: 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), pp. 1603–1607. IEEE (2017)
6. Cho, P.W., Goldrick, M., Smolensky, P.: Incremental parsing in a continuous dynamical system: sentence processing in gradient symbolic computation. Linguistics Vanguard **3**(1), 1–10 (2017)
7. Chollet, F., et al.: Keras (2015). https://keras.io

8. Demidovskij, A.: Implementation aspects of tensor product variable binding in connectionist systems. In: Bi, Y., Bhatia, R., Kapoor, S. (eds.) IntelliSys 2019. AISC, vol. 1037, pp. 97–110. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-29516-5_9
9. Demidovskij, A.: Automatic construction of tensor product variable binding neural networks for neural-symbolic intelligent systems. In: Proceedings of 2nd International Conference on Electrical, Communication and Computer Engineering, pp. not published, accepted. IEEE (2020)
10. Demidovskij, A., Babkin, E.: Developing a distributed linguistic decision making system. Business Informatics 13(1 (eng)) (2019)
11. Demidovskij, A., Babkin, E.: Towards designing linguistic assessments aggregation as a distributed neuroalgorithm. In: 2020 XXII International Conference on Soft Computing and Measurements (SCM)), pp. not published, accepted. IEEE (2020)
12. Demidovskij, A.V.: Towards automatic manipulation of arbitrary structures in connectivist paradigm with tensor product variable binding. In: Kryzhanovsky, B., Dunin-Barkowski, W., Redko, V., Tiumentsev, Y. (eds.) NEUROINFORMATICS 2019. SCI, vol. 856, pp. 375–383. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-30425-6_44
13. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
14. Fodor, J.A., Pylyshyn, Z.W., et al.: Connectionism and cognitive architecture: a critical analysis. Cognition **28**(1–2), 3–71 (1988)
15. Gallant, S.I., Okaywe, T.W.: Representing objects, relations, and sequences. Neural Comput. **25**(8), 2038–2078 (2013)
16. Golmohammadi, D.: Neural network application for fuzzy multi-criteria decision making problems. Int. J. Prod. Econ. **131**(2), 490–504 (2011)
17. Huang, Q., Smolensky, P., He, X., Deng, L., Wu, D.: Tensor product generation networks for deep NLP modeling. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1 (Long Papers), pp. 1263–1273. Association for Computational Linguistics, New Orleans (2018). https://doi.org/10.18653/v1/N18-1114. https://www.aclweb.org/anthology/N18-1114
18. Legendre, G., Miyata, Y., Smolensky, P.: Distributed recursive structure processing. In: Advances in Neural Information Processing Systems, pp. 591–597 (1991)
19. Leitgeb, H.: Interpreted dynamical systems and qualitative laws: from neural networks to evolutionary systems. Synthese **146**(1–2), 189–202 (2005)
20. McCoy, R.T., Linzen, T., Dunbar, E., Smolensky, P.: RNNS implicitly implement tensor product representations. arXiv preprint arXiv:1812.08718 (2018)
21. Palangi, H., Smolensky, P., He, X., Deng, L.: Question-answering with grammatically-interpretable representations. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
22. de Penning, H.L.H., Garcez, A.S.d., Lamb, L.C., Meyer, J.J.C.: A neural-symbolic cognitive agent for online learning and reasoning. In: Twenty-Second International Joint Conference on Artificial Intelligence (2011)
23. Pinkas, G.: Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge. Artif. Intell. **77**(2), 203–247 (1995)
24. Pinkas, G., Lima, P., Cohen, S.: A dynamic binding mechanism for retrieving and unifying complex predicate-logic knowledge. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012. LNCS, vol. 7552, pp. 482–490. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33269-2_61

25. Pinkas, G., Lima, P., Cohen, S.: Representing, binding, retrieving and unifying relational knowledge using pools of neural binders. Biol. Inspired Cogn. Archit. **6**, 87–95 (2013)
26. Serafini, L., Garcez, A.d.: Logic tensor networks: deep learning and logical reasoning from data and knowledge. arXiv preprint arXiv:1606.04422 (2016)
27. Smolensky, P.: Tensor product variable binding and the representation of symbolic structures in connectionist systems. Artif. Intell. **46**(1–2), 159–216 (1990)
28. Smolensky, P., Goldrick, M., Mathis, D.: Optimization and quantization in gradient symbol systems: a framework for integrating the continuous and the discrete in cognition. Cogn. Sci. **38**(6), 1102–1138 (2014)
29. Smolensky, P., Legendre, G.: The Harmonic Mind: From Neural Computation to Optimality-theoretic Grammar (Cognitive Architecture), Vol. 1. MIT press, Cambridge (2006)
30. Soulos, P., McCoy, T., Linzen, T., Smolensky, P.: Discovering the compositional structure of vector representations with role learning networks. arXiv preprint arXiv:1910.09113 (2019)
31. Teso, S., Sebastiani, R., Passerini, A.: Structured learning modulo theories. Artif. Intell. **244**, 166–187 (2017)
32. Wei, C., Liao, H.: A multigranularity linguistic group decision-making method based on hesitant 2-tuple sets. Int. J. Intell. Syst. **31**(6), 612–634 (2016)
33. Yousefpour, A., et al.: Failout: achieving failure-resilient inference in distributed neural networks. arXiv preprint arXiv:2002.07386 (2020)