Cognitive Systems Research 71 (2022) 52–63

Contents lists available at ScienceDirect

Cognitive Systems Research

journal homepage: www.elsevier.com/locate/cogsys

Vector Semiotic Model for Visual Question Answering

Alexey K. Kovalev [a,b], Makhmud Shaban [b], Evgeny Osipov [c], Aleksandr I. Panov [a,d,*]

[a] Artificial Intelligence Research Institute FRC CSC RAS, Moscow, Russia
[b] HSE University, Moscow, Russia
[c] Lulea University of Technology, Lulea, Sweden
[d] Moscow Institute of Physics and Technology, Moscow, Russia

ARTICLE INFO

Keywords:
Vector-symbolic architecture
Semiotic approach
Symbol grounding problem
Causal network
Visual Question Answering

ABSTRACT

In this paper, we propose a Vector Semiotic Model as a possible solution to the symbol grounding problem in the context of Visual Question Answering. The Vector Semiotic Model combines the advantages of a Semiotic Approach implemented in the Sign-Based World Model and Vector Symbolic Architectures. The Sign-Based World Model represents information about a scene depicted on an input image in a structured way and grounds abstract objects in an agent's sensory input. We use the Vector Symbolic Architecture to represent the elements of the Sign-Based World Model on a computational level. Properties of a high-dimensional space and operations defined for high-dimensional vectors allow encoding the whole scene into a high-dimensional vector with the preservation of the structure. That leads to the ability to apply explainable reasoning to answer an input question. We conducted experiments are on a CLEVR dataset and show results comparable to the state of the art. The proposed combination of approaches, first, leads to the possible solution of the symbol-grounding problem and, second, allows expanding current results to other intelligent tasks (collaborative robotics, embodied intellectual assistance, etc.).

1. Introduction

Visual question answering (VQA) is one of the most challenging Artificial Intelligence (AI) tasks that attracts many researchers (Manmadhan & Kovoor, 2020; Wu, et al., 2017). The task is to predict an answer given an input image and a question about this image. VQA is the basis for several complex AI applications relying on automatic understanding of both off-line and real-time video streams, including an assistant for visually impaired people (Gurari, et al., 2019), collaborative robotics, and embodied intellectual assistants (robots). VQA is a multi-modal task requiring the interplay of fine-grained image analysis techniques and advanced natural language models. The major challenge to solve in VQA is the symbol grounding problem (Besold & Kühnberger, 2015; Harnad, 1990; Osipov, 2015; Steels, 2008), which is a fundamental but not yet fully resolved AI problem. In short, the problem is how to relate symbols in an AI model to objects and situations in real life. In the case of VQA, the problem is translated into finding the mapping between the symbols of a natural language processing (NLP) model for interpreting questions to the objects and situations in a visual scene processed with a given computer vision model. Apart from the fact that VQA and QA systems attract the attention of researchers, their practical, highly specialized application is also of interest (Bongini, Becattini, Bagdanov, & Bimbo, 2020; Butt, Ashraf, Siddiqui, Sidorov, & Gelbukh, 2021; Gupta, Suman, & Ekbal, 2021; He, Zhu, Zhang, Chen, & Caverlee, 2020; Lobry, Marcos, Murray, & Tuia, 2020; Manna, Das, & Gelbukh, 2021; Pathak, Manna, Partha Pakray, Gelbukh, & Bandyopadhyay, 2021; Singh & Shekhar, 2020; Su, et al., 2020; Vo, Phung, & Ly, 2020; Yu, Wu, et al., 2020; Zhang, Deng, Ma, & Lam, 2020).

This paper contributes by proposing a possible solution to the symbol grounding problem by co-modeling the natural language and visual input using Vector Symbolic Architectures (VSA) as a common information representation technique along the entire VQA pipeline. This paper contributes by proposing a possible solution to the symbol grounding problem by co-modeling the natural language and visual input using Vector Symbolic Architectures (VSA) as a common information representation technique along the entire VQA pipeline. To the best of our knowledge, the proposed model is the first attempt to solve the VQA benchmark dataset as CLEVR with Vector Symbolic Architectures. VSA (Gayler, 1998b), also known as Hyperdimensional computing (Kanerva, 2009), is a family of models for representing and manipulating data in a high-dimensional space, originally proposed in Cognitive Psychology and Cognitive Neuroscience as connectionist models for symbolic reasoning. The proposed solution implements the Semiotic Approach (Roy, 2005; Trifonas, 2015) that allows tracing

* Corresponding author at: Moscow Institute of Physics and Technology, Moscow, Russia.
E-mail address: panov.ai@mipt.ru (A.I. Panov).

https://doi.org/10.1016/j.cogsys.2021.09.001
Received 5 August 2021; Received in revised form 10 September 2021; Accepted 16 September 2021
Available online 26 October 2021

the question answering process, which is infeasible in purely neural network models. The model is specifically tailored for implementation on resource-constrained devices since the inference path of the pipeline is implemented using integer vector operations only. The trade-off between the model accuracy and implementation complexity is controlled by appropriately tuning the dimensionality of the vector representation. We support our findings by demonstrating that our model performance matches the state of the art on the standard VQA dataset — CLEVR (Johnson, et al., 2017).

The paper is organized as follows. Section 2 discusses VQA and outlines the related solutions. The theories behind the proposed solution are outlined in Section 3. We describe the main contribution, i.e., the usage of VSA and the semiotic approach in the VQA pipeline, in Section 4. Section 5 presents the evaluation results of the proposed solution on the CLEVR dataset. We discuss our findings in Section 6. The final conclusion appears in Section 7.

2. Visual question answering in the scope of related works

In VQA, the goal is to answer questions in natural language given an image of interest. A typical VQA scenario in the area of robotics could be ordering a robot to *"bring the red box"* and further specify the location of the box as *"the box to the left of the sofa"*. The approaches for solving VQA tasks fall into three main categories: (1) purely neural network approaches (Anderson, et al., 2017; Chang, Yang, Park, & Kwak, 2018; Kim, Jun, & Zhang, 2018; Li, Yatskar, Yin, Hsieh, & Chang, 2019; Lu, Batra, Parikh, & Lee, 2019; Ma, Lu, & Li, 2015; Malinowski, Rohrbach, & Fritz, 2015; Santoro, et al., 2017; Su, Zhu, et al., 2020; Tan & Bansal, 2019; Yu, Yu, Cui, Tao, & Tian, 2019; Zhou, Tian, Sukhbaatar, Szlam, & Fergus, 2015), including those that use attention mechanisms of different kinds (Anderson, et al., 2017; Kim et al., 2018; Yu et al., 2019) or a transformer architecture (Li et al., 2019; Lu et al., 2019; Su, Zhu, et al., 2020; Tan & Bansal, 2019); (2) approaches using different additional external sources such as knowledge bases, databases, or ontology for reasoning (Vo et al., 2020; Wu, Shen, Wang, Dick, & v. d. Hengel, 2018; Wu, Wang, Shen, Dick, & Van Den Hengel, 2016; Yu, et al., 2020); (3) solutions based on a neural-symbolic approach that combines neural network models with models that perform interpretable computations (Andreas, Rohrbach, Darrell, & Klein, 2016; Mao, Gan, Kohli, Tenenbaum, & Wu, 2019; Yi, et al., 2018).

The neural-symbolic approach is of particular interest. It uses connectionist models to work with raw data and symbolic computation for interpretable reasoning. VSA is a suitable tool to bridge the gap between connectionist and symbolic approaches as it treats numeric vectors as symbols. There are works that apply VSA to the simplified VQA task (Montone, O'Regan, & Terekhov, 2017)or tasks closely related to VQA (Kleyko, Osipov, Gayler, Khan, & Dyer, 2015; Yilmaz, 2015). In Montone et al. (2017), the authors used a simple synthesized dataset with 2D scenes. Each scene contains two geometric figures of four different shapes and colors. Figures can be in four different positions in the image. The dataset contains all possible combinations of those features (3072 images in total). A feedforward network with two layers is used to predict the VSA description of the given image. The authors used five question templates to query the system. The model showed satisfactory results, but the simplicity of the test environment makes it difficult to generalize these results to more complex tasks. In Kleyko et al. (2015), VSA is used to represent stimuli, i.e., pattern and color groups, in a maze in which honey bees are trained. The proposed episode encoding is suitable for VQA as it takes into account objects and their relationships. In Yilmaz (2015), the authors use VSA operations for logical inference on HD representations of images from the CIFAR-10 (Krizhevsky, 2009) dataset. The HD representations are obtained from an autoencoder hidden layer by binarizing and feeding them to a cellular automaton. The cellular automaton evolution is computed following the chosen rule. Concatenation of several automaton states serves as a high-dimensional vector. The logical reasoning performs by a series of VSA operations.

The symbol grounding problem (Harnad, 1990) is addressed in Chen, Vondrick, and Lipson (2021), Schmidtke (2021a, 2021b), Sheppard and Lohan (2020) and Talbot, Dayoub, Corke, and Wyeth (2020). In Chen et al. (2021), an observer (one agent) attempts to model the behavior of an actor (another agent) only through visual data without any prior symbolic information, by which the authors try to avoid the symbol grounding problem. In Schmidtke (2018, 2021a, 2021b) the abstract symbol grounding problem is discussed (how symbols that arise as a result of reasoning or received from other agents can be grounded in the environment of different nature) and proposed a solution based on the atomic Context Logic and the Activation Bit Vector Machine (a variation of VSA). In Talbot et al. (2020) a navigation system proposed that uses a special data structure called abstract map that allows the system to utilize the symbolic spatial information and raw, low-level sensorimotor measurements.

The state-of-the-art solution we chose as a baseline is the NS-VQA (Yi, et al., 2018) model. This model falls into the neural-symbolic paradigm. NS-VQA processes visual and textual information separately. The model uses neural networks to detect objects on a scene and extract their attributes and coordinates. Then this is used to construct a tabular (symbolic) representation of the scene. An encoder–decoder model translates an input question to a sequence (a program) of elementary functions. Each function either performs simple filtering, comparison, logical, set, or other auxiliary operations. The program executor performs deterministic symbolic reasoning by applying the program to the tabular scene representation to obtain an answer.

Our solution relies upon the preliminary result from Kovalev, Panov, and Osipov (2020). In Kovalev et al. (2020), the authors propose a model that combines VSA and a semiotic approach. They demonstrate the ability of such a combination to solve a simple VQA task. The semiotic approach rests on the Sign-Based World Model (SBWM) proposed in Osipov, Panov, and Chudova (2014), Panov (2017). SBWM is a cognitive architecture that allows modeling cognitive tasks, i.e., hierarchical planning for one agent (Aitygulov, Kiselev, & Panov, 2018; Kiselev, Kovalev, & Panov, 2018; Kiselev & Panov, 2019), a group of agents (Kiselev & Panov, 2017), goal setting (Panov, 2019), and reasoning (Kiselev et al., 2018; Kovalev & Panov, 2019). The approach founds on a uniform representation of objects, actions, and situations as a **sign**.

3. Outline of the core theories: Vector Symbolic Architectures and the semiotic approach

3.1. Vector symbolic architectures

Vector Symbolic Architectures, also known as hyperdimensional computing (Kanerva, 2009), is a family of bio-inspired methods of representing and manipulating concepts and their meanings in a high-dimensional space. Vectors of high (but fixed) dimensionality (denoted as d) are the basis for representing information in hyperdimensional computing. These vectors are often referred to as high-dimensional vectors or HD vectors. The information is distributed across HD vector positions, therefore, HD vectors use distributed representations. Distributed representations are contrary to the localist representations (which are conventionally used for computations) since any subset of the positions can be interpreted. In other words, a particular position of an HD vector does not have any interpretable meaning. Only the whole HD vector can be interpreted as a holistic representation of some entity, which, in turn, bears some information load. In this paper, the object's attributes in a scene are the atomic symbols of a system, and their HD vectors are generated randomly. Atomic HD vectors are stored in the so-called item memory (IM), which in its simplest form is a matrix. Denote the item memory as $\mathbf{H}$, where $\mathbf{H} \in [d \times a]$, and a is a number of stored

vectors. For a given symbol S its corresponding HD vector from $\mathbf{H}$ is denoted as $\mathbf{H}_S$. For the simplicity of the presentation, we describe the functionality of VSA for the case of bipolar vectors ($\mathbf{H}_S \in \{-1,+1\}^{[d\times 1]}$) randomly with equal probabilities for $+1$ and -1. To encode the symbol S as the vector $\mathbf{H}_S$, we generate a random HD vector (draw a sample from a vector space $\{-1,+1\}^{[d\times 1]}$) and store it in the item memory $\mathbf{H}$. An important property of high-dimensional spaces is that with an extremely high probability all random HD vectors are dissimilar to each other (quasi orthogonal) (Kleyko, et al., 2021).

In order to manipulate atomic HD vectors, hyperdimensional computing defines operations and a similarity measure on HD vectors. In this paper, we use the cosine similarity for characterizing the similarity. Three key operations for computing with HD vectors are bundling, binding, and permutation.

The binding operation is used to bind two HD vectors together. The result of binding is another HD vector. For example, for two symbols S_1 and S_2 the result of the binding of their HD vectors (denotes as **b**) is calculated as: $\mathbf{b} = \mathbf{H}_{S_1} \odot \mathbf{H}_{S_2}$, where the notation $\odot$ for the Hadamard product is used to denote the binding operation since this paper uses positionwise multiplication for binding. An important property of the binding operation is that the resultant HD vector **b** is dissimilar to the HD vectors being bound, i.e., the cosine similarity between **b** and $\mathbf{H}_{S_1}$ or $\mathbf{H}_{S_2}$ is approximately 0.

An alternative approach to binding, when there is only one HD vector, is to permute (rotate) the HD vector positions. It is convenient to use a fixed permutation (denoted as ρ) to bind the position of a symbol in a sequence to an HD vector representing the symbol in that position. Thus, for a symbol S_1 the result of permutation of its HD vector (denoted as **r**) is calculated as: $\mathbf{r} = \rho(\mathbf{H}_{S_1})$. Similar to the binding operation, the resultant HD vector **r** is dissimilar to $\mathbf{H}_{S_1}$.

The last operation is bundling. It is denoted by + and implemented via a positionwise addition with a threshold. If the summation result exceeds the threshold, then it is replaced by the threshold value. The bundling operation combines several HD vectors into a single HD vector. For example, for S_1 and S_2 the result of the bundling of their HD vectors (denoted as **a**) is simply: $\mathbf{a} = \mathbf{H}_{S_1} + \mathbf{H}_{S_2}$. In contrast to binding and permutation operations, the resultant HD vector **a** is similar to all bundled HD vectors, i.e., the cosine similarity between **b** and $\mathbf{H}_{S_1}$ or $\mathbf{H}_{S_2}$ is more than 0.

An important application of a combination of bundling and binding is when we represent an object as a collection of attribute-value pairs. Consider an example where the object has two attributes A_1, A_2 hat may take the values V_{A_1} and V_{A_2} correspondingly. We can represent the object as a vector $\mathbf{o} = \mathbf{H}_{A_1} \odot \mathbf{H}_{V_{A_1}} + \mathbf{H}_{A_2} \odot \mathbf{H}_{V_{A_2}}$. When we want to get the value of a particular attribute, we apply unbinding operation (binding with desirable attribute HD vector) $\mathbf{o} \odot \mathbf{H}_{A_1} = \mathbf{H}_{A_1} \odot \mathbf{H}_{V_{A_1}} \odot \mathbf{H}_{A_1} + \mathbf{H}_{A_2} \odot \mathbf{H}_{V_{A_2}} \odot \mathbf{H}_{A_1} = \mathbf{H}_{V_{A_1}} + noise = \tilde{\mathbf{H}}_{V_{A_1}}$ and get a noisy version $\tilde{\mathbf{H}}_{V_{A_1}}$ of an HD vector $\mathbf{H}_{V_{A_1}}$. Then we can find a clean version of a vector $\mathbf{H}_{V_{A_1}}$ by searching an item memory $\mathbf{H}$ for a vector closest to the noisy vector $\tilde{\mathbf{H}}_{V_{A_1}}$.

While basic VSA operations were presented above for bipolar random vectors, other numerical systems are also used for constructing VSA. HD vectors can be with real, complex, or binary components. These flavors of VSA come under many different names: Holographic Reduced Representation (HRR) (Plate, 2003), Multiply-Add-Permute (MAP) (Gayler, 1998a; Gayler & Wales, 1998), Binary Spatter Codes (Kanerva, 1997), Sparse Binary Distributed Representations (SBDR) (Rachkovskij, 2001), Sparse Block-Codes (Laiho, Poikonen, Kanerva, & Lehtonen, 2015). All these different models target specific computing hardware but have similar computational properties. In what follows we use MAP (Gayler, 1998a; Gayler & Wales, 1998) VSA on dense bipolar vectors. In Schlegel, Neubert, and Protzel (2020), chosen VSA is denoted as MAP-B. It provides commutative and associative binding and unbinding compare to such variants as HRR. It is also more computationally efficient than HRR as it uses only integer numbers but provides more capacity than BSC that uses binary vectors.

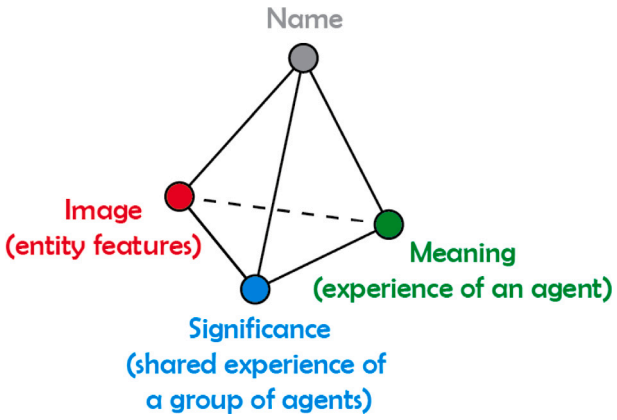


Fig. 1. Sign is the main component of the Sign-Based World Model cognitive architecture. This four-component structure is used to represent the agent's knowledge about the environment, other agents, and itself.

3.2. Sign-based world model

The Sign-Based World Model (SBWM) (Osipov et al., 2014; Panov, 2019) is a framework for modeling cognitive tasks. It relies on the concept of a sign representing the agent's knowledge about the environment it operates in, other agents it interacts with, and itself. The signs are organized in a hierarchical semantic network, which is called a *semiotic network*. Conceptually, the sign is a four-component structure illustrated in Fig. 1. The four components are image, meaning, significance, and name. They represent different aspects of the agent's knowledge; the meaning component implies the agent's experience; the significance component stands for commonsense knowledge; the image component is used to distinguish signs; the name possesses a nominative function. The sign components by themselves compose semantic networks on meanings, significances, images, and names.

Depending on the task solving one component of the sign may play a more or less important role than the others. For example, in solving planning tasks (Aitygulov et al., 2018; Kiselev et al., 2018; Kiselev & Panov, 2019), role distribution tasks (Kiselev & Panov, 2017) in a group of agents, or goal setting (Panov, 2019) significance and meaning components come to the fore since it is necessary to take into account the agents' experience and the rules of the environment. In a reasoning task (Kiselev et al., 2018; Kovalev & Panov, 2019), the significance and image components play the main role. In this paper, we use only the name and image components of the sign to demonstrate the ability of SBWM with VSA to solve the VQA task. The name component links symbols to corresponding words in the question. In the context of the symbol grounding problem and VQA, the image component, which implements the recognition function and allows an association of the signs with the agent sensor outputs, plays a crucial role. The questions in the dataset ask about external features, the number of objects, or their relationships. The network of images encodes that information, so it is necessary and sufficient to answer such questions. To answer other types of questions, for example, "What is this object used for?", we have to use other causal networks, the network on significances in this case.

Let us discuss a causal matrix — the structure representing the significance, meaning, and image components of a sign on a low level (the name component is represented as a finite string over a finite alphabet).

A causal matrix z is defined as a tuple of length t of events e_i: $z = \langle e_1, e_2, \dots, e_t \rangle$. Each event e_i represents the appearance of a particular feature f_j at time step i and is a binary vector of length h. The nature of the feature f_j may be different and depends on the type of the sign component but in the general case, it is represented by a tuple of causal

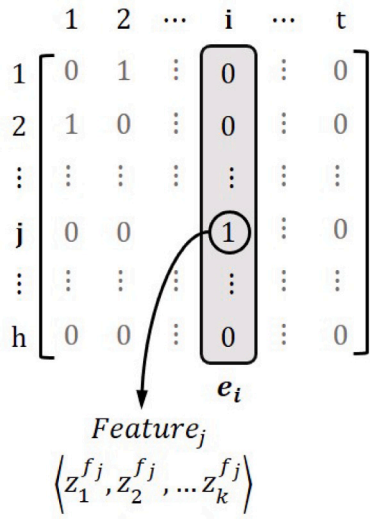


Fig. 2. Causal matrix is a binary matrix $[h \times t]$. The event e_i corresponds to the ith column of the matrix. The 1 in the position z_{ji} means that a *feature* f_j appeared in the event e_i. In the general case, a *feature* f_j is a tuple of causal matrices $z_1^{f_j}, z_2^{f_j}, \dots, z_k^{f_j}$.

matrices that form a causal tensor. Thus a causal matrix is a binary h by t matrix (Fig. 2).

The 1 in the position z_{ji} in a causal matrix serves as a link to other matrices that correspond to the feature f_j and means that the feature f_j is included in the corresponding component of the sign. Thereby causal matrices are organized into a hierarchical semantic network where the tuples of causal matrices are the nodes and the links are the relations between these tuples.

Several features may appear in the same event consequently, several 1 appear in the corresponding column that complicates the hierarchical structure. But without loss of generality, we can bring matrices in a form where each column contains only one filled position by adding new levels of hierarchy.

The event index t may serve as a discrete time whenever we represent entities with a dynamic nature. In the scope of an image component, it does not matter in what sequence the features appear. It follows that causal matrices in the tuple differ only in the permutation of the columns, which in the worst case will lead to the need to store $t!$ matrices for an image component. Further, we show how HD representation helps eliminate this need and leaves with the only t of them.

In this paper, we use a causal network built on images (the simplified version is shown in Fig. 3) to represent the scenes in a structured way.

3.3. Vector Semiotic Model: The realization of the Sign-Based World Model using Vector Symbolic Architectures and an approach to symbol grounding

In this section, we propose the Vector Semiotic Model (VSM), in which we use VSA operations to represent SBWM structures as HD vectors. We also show how this representation relates to symbol grounding. We use **bold font** to denote an HD vector of a corresponding SBWM structure (typed in *italics*) and **H** with a corresponding subscript to show that this HD vector is stored in the item memory.

We can represent a causal matrix as a set of events and use bundling to construct an HD representation from separate events (Fig. 4). First, we have to map every event e_i to a corresponding HD vector $\mathbf{H}_{e_i}$ and then apply bundling to the collection of vectors $\mathbf{H}_{e_1}, \mathbf{H}_{e_2}, \dots, \mathbf{H}_{e_t}$:

$$\mathbf{z} = \sum_{i=1}^{t} \mathbf{H}_{e_i} \tag{1}$$

In the case of the image component, the mapping to the HD vector solves the problem of storing $t!$ causal matrices for each sign as HD representation does not consider the order of events.

To represent a link from a causal matrix z_1 to a causal matrix z_2 (Fig. 5) we, first, transform z_2 to an HD vector $\mathbf{z}_2$, second, split z_1 into events $e_i^{z_1}$, and map them to HD vectors $\mathbf{H}_{e_i^{z_1}}$, and then bind $\mathbf{z}_2$ with a corresponding HD vector $\mathbf{H}_{e_i^{z_1}}$:

$$\mathbf{z}_1 = \sum_{i=1}^{t} \mathbf{H}_{e_i^{z_1}} \odot \mathbf{z}_i \tag{2}$$

Thus, the scene description on a causal network of images (Fig. 3) is represented as an HD vector preserving a hierarchical structure and grounding each current level of a hierarchy in all the previous levels up to a sensory input:

$$\mathbf{z}_{scene} = \sum_{i=1}^{h} \mathbf{H}_{z_{object_i}} \odot \sum_{j=1}^{k} \mathbf{H}_{z_{attr_j}} \odot \mathbf{H}_{z_{val_j}} \tag{3}$$

where *attr* states for the attributes (*Size*, *Shape*, *Color*, *Material*, etc.) and *val* states for the attribute values (*Large*, *Cube*, *Red*, *Rubber*, etc.).

3.4. Combination of the semiotic approach and Vector Symbolic Architecture provides a possible solution to the symbol grounding problem

Recall that we use only the part of SBWM, i.e., a causal network built on images, to represent the scene and answer questions. The causal network allows representing scenes in a structured way. A scene is viewed as a collection of objects and an object as a collection of attribute-value pairs. The name component of the sign plays an auxiliary role, linking the sign and the corresponding word from the question. The symbol grounding in the vector semiotic model comes due to the two properties, i.e., the hierarchical nature of the causal network and the fact that the lowest level image component is mapped to the raw input signal. Thus, the entire scene and every included causal matrix are grounded in the sensory input stream.

From the perspective of VSA, each symbol – the word in the context of VQA – is a high-dimensional vector. Therefore, we can operate on these symbols using vector operations and easily switch between representations using the item memory **H**. In SBWM, each sign has a name and, in the context of VQA, the name of the sign corresponds to the name of the sign image component. Thus we can connect separate vectors from the item memory **H** (which are stored in it in an unstructured form) using a causal network on image components into a structure defined by SBWM.

Thus, the joint use of approaches allows not only grounding symbols in the agent's sensory inputs. It also allows operating directly with symbols in their vector representation, preserving connections to these inputs.

4. Visual question answering with Vector Semiotic Model

In this section we present our main contribution — the solution to the VQA task using the Vector Semiotic Model presented in the previous section.

4.1. Encoding spatial relations via high-dimensional vectors

Assume the situation when the agent is asked a question like *"What is the object to the right of the red circle?"*. To answer such kind of questions the system have to have a notion of *"to the right"*. The simplest way to obtain such a notion is to exploit the exact coordinates of objects in terms of pixels extracted directly from the image. Then just by comparing the coordinates of objects, integer numbers, the system can answer questions of such types as *"to the right/left"*, *"above/below"* etc. Thus, the next step is to generate an HD vector for the coordinates for each object. Wherein the process of generation must preserve ordering

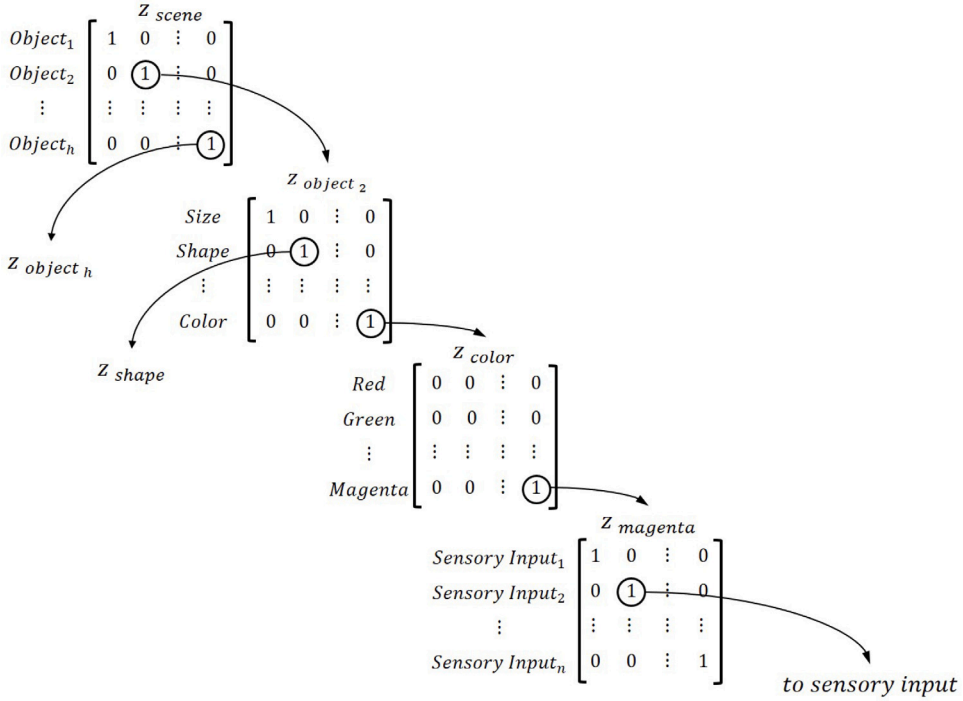


Fig. 3. Simplified version of a scene representation on a causal network of images. The scene is viewed as a collection of objects, and an object is a collection of attribute-value pairs. The lowest level of the hierarchy is mapped to the sensory input of an agent. z denotes a causal matrix.

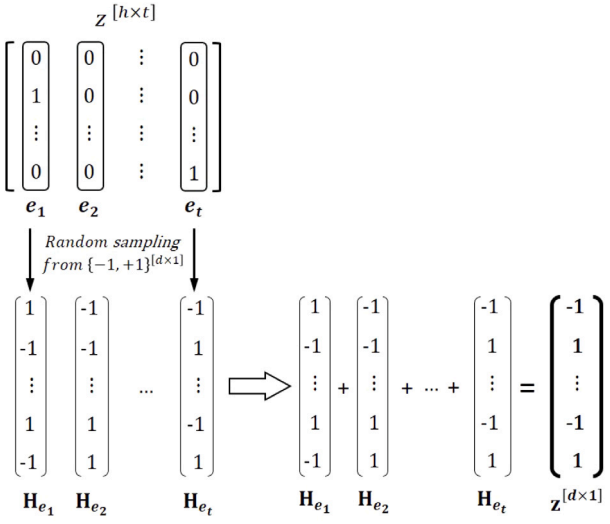


Fig. 4. Mapping from a causal matrix z to an HD vector $\mathbf{z}$. First, a matrix z is split into events e_i. Second, for each event e_i a corresponding HD vector $\mathbf{H}_{e_i}$ is sampled from $\{-1,+1\}^{[d\times 1]}$. Third, vectors $\mathbf{H}_{e_i}$ are bundled to form a resulting vector $\mathbf{z}$.

corresponding to integer representation. Let us consider that an image depicted in Fig. 6 is shown to the system and the question *"What is the object to the right of the red cylinder?""* is asked and let our system scan the picture from left to right and from the top to bottom.

The objects on the image are detected in the following order: a yellow cylinder, a red cylinder, and a cube. The coordinates of the objects are sorted in ascending order. The left-most x coordinate viewed x_0 and the top-most coordinate y as y_0. We are not interested in the exact values of x_0 and y_0 but we associate them with random HD vectors $\mathbf{H}_{x_0}$ and $\mathbf{H}_{y_0}$ respectively. Then the coordinates of the yellow cylinder are (x_0, y_2) and for the red cylinder they are (x_1, y_0). We want to map every coordinate to an HD vector, and this mapping must preserve ordering. It can be obtained by using a special case of the permutation operation – a circular shift. Let $\rho^n(\mathbf{H})$ be a vector obtained from vector $\mathbf{H}$ by circular shifting its components to the right by n positions. Then we can encode y_1 as $\mathbf{y}_1 = \rho^1(\mathbf{H}_{y_0})$. Following this procedure, we obtain HD vectors for all the object coordinates in the image:

1. Yellow Cylinder (YC): $x_0 := \mathbf{H}_{x_0}$, $y_2 := \rho^2(\mathbf{H}_{y_0})$
2. Red Cylinder (RC): $x_1 := \mathbf{x}_1 = \rho^1(\mathbf{H}_{x_0})$, $y_0 := \mathbf{H}_{y_0}$
3. Cube (C): $x_2 := \mathbf{x}_2 = \rho^2(\mathbf{H}_{x_0})$, $y_1 := \rho^1(\mathbf{H}_{y_0})$

For each object in an image, an HD vector is sampled and stored in the item memory, these vectors are used as identifiers and denoted as $\mathbf{H}_{YC}$, $\mathbf{H}_{RC}$, $\mathbf{H}_C$. The item memory, a vector corresponding to the color attribute $\mathbf{H}_{color}$ (we assume that thr objects have only one attribute for simplicity), vectors corresponding to its values $\mathbf{H}_{yellow}$, $\mathbf{H}_{red}$, $\mathbf{H}_{blue}$, and vectors $\mathbf{H}_x$, $\mathbf{H}_y$ that are identifiers for the coordinates are also stored.

The full description of the scene depicted in Fig. 6 will be:

$$\begin{aligned}\mathbf{z}_{scene} = {} & \mathbf{H}_{YC} \odot [\mathbf{H}_{color} \odot \mathbf{H}_{yellow}] + \mathbf{H}_{RC} \odot [\mathbf{H}_{color} \odot \mathbf{H}_{red}] \\ & + \mathbf{H}_C \odot [\mathbf{H}_{color} \odot \mathbf{H}_{blue}] + \\ & \mathbf{H}_x \odot [\mathbf{H}_{YC} \odot \mathbf{H}_{x_0} + \mathbf{H}_{RC} \odot \mathbf{x}_1 + \mathbf{H}_C \odot \mathbf{x}_2] + \\ & \mathbf{H}_y \odot [\mathbf{H}_{YC} \odot \mathbf{y}_2 + \mathbf{H}_{RC} \odot \mathbf{H}_{y_0} + \mathbf{H}_C \odot \mathbf{y}_1] \end{aligned} \quad (4)$$

4.2. Complex operations on a vector scene representation

In this section, we explore complex operations on a vector representation of a scene that we will use to answer questions. There are three main types of complex operations on HD representation, first, how to obtain an arbitrary attribute value of a given object, second, how to obtain an object with an exact attribute value, and third, how to find objects that are left to, right to, behind, or in front of a given object. For simplicity, we assume that exactly one object in the scene has a queried attribute value.

When we ask to obtain an object with value $\mathbf{H}_{val}$ of the attribute $\mathbf{H}_{attr}$ on the scene $\mathbf{z}_{scene}$ we have to perform three operations:

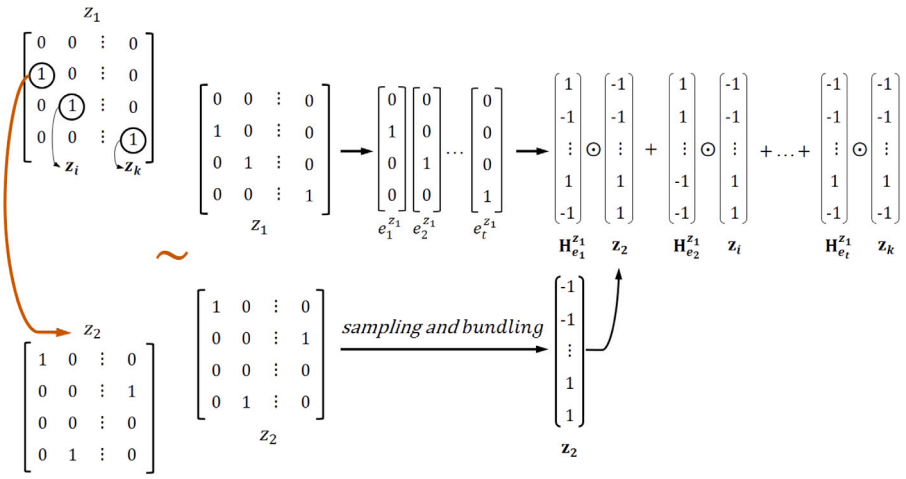


Fig. 5. Representation of a link from a causal matrix z_1 to a causal matrix z_2. First, a matrix z_2 is represented as an HD vector $\mathbf{z}_2$. Second, a matrix z_1 is split into events e_i and HD vectors $\mathbf{H}_{e_i}^{z_1}$ are sampled for each event. Then, a vector $\mathbf{z}_2$ is bound with a corresponding $\mathbf{H}_{e_i}^{z_1}$ vector and added to the bundle to form an HD vector $\mathbf{z}_1$.

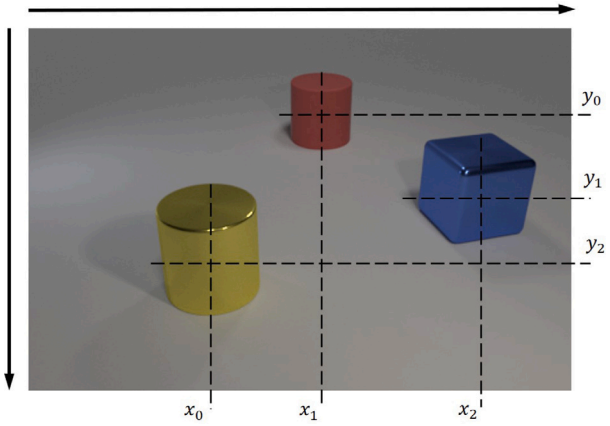


Fig. 6. Example image from the CLEVR training split. To illustrate the spatial relation encoding we consider only the x and y axes, and assume that the objects have only a color attribute. An agent perceives the image left to right and from top to bottom. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

1. Construct a temporary vector of an attribute-value pair:

   $\mathbf{t} = \mathbf{H}_{attr} \odot \mathbf{H}_{val}$;

2. Obtain a noisy version of the object vector: $\tilde{\mathbf{H}}_{object} = \mathbf{z}_{scene} \odot \mathbf{t}$;
3. Pass $\tilde{\mathbf{H}}_{object}$ vector through the item memory, find the closest vector and return it as a clean vector $\mathbf{H}_{object}$ for the object.

If there are objects with the same attribute values, instead of searching for the closest vector in step 3, we could return a list of object vectors with distance to the query vector less than the specified threshold thr.

The detailed version of this algorithm that works for several objects with the same attribute value is represented in Algorithm 1.

The procedure $get\_value(\mathbf{z}_{scene}, attribute, object)$ that returns a value of a given object attribute is:

1. Construct a temporary vector of an object-attribute pair:

   $t = \mathbf{H}_{object} \odot \mathbf{H}_{attr}$;

2. Obtain a noisy version of the value vector: $\tilde{\mathbf{H}}_{val} = \mathbf{z}_{scene} \odot \mathbf{t}$;
3. Pass $\tilde{\mathbf{H}}_{val}$ vector through the item memory, find the closest vector and return it as a clean vector $\mathbf{H}_{val}$ for the value.

Algorithm 1: Get objects with a specific attribute value.

1: **procedure** GET_OBJECTS($\mathbf{z}_{scene}, attribute, value, thr$)
2: Get HD vector $\mathbf{H}_{attr}$ for an *attribute* from $\mathbf{H}$
3: Get HD vector $\mathbf{H}_{val}$ for a *value* from $\mathbf{H}$
4: Assign a *value* to an *attribute*: $\mathbf{t} = \mathbf{H}_{attr} \odot \mathbf{H}_{val}$
5: Get a noisy bundle with objects: $\tilde{\mathbf{H}}_{objects} = \mathbf{z}_{scene} \odot \mathbf{t}$
6: Query $\mathbf{H}$ with a vector $\tilde{\mathbf{H}}_{objects}$
7: **if** $sim(\tilde{\mathbf{H}}_{objects}, \mathbf{H}_{object_i}) > thr$ **then** ▷ $sim()$ - a similarity metric
8: Add $\mathbf{H}_{object_i}$ to a list of objects *result*
9: Return *result*

With these procedures, we can answer such simple questions as *"Is there an object with a given attribute value?"* or *"What is the attribute value of a given object?"*. The detailed version of this algorithm is represented in Algorithm 2.

Algorithm 2: Get the value of a specific attribute of a given object.

1: **procedure** GET_VALUE($\mathbf{z}_{scene}, attribute, object$)
2: Get HD vector $\mathbf{H}_{attr}$ for an *attribute* from $\mathbf{H}$
3: Get HD vector $\mathbf{H}_{object}$ for an *object* from $\mathbf{H}$
4: Assign *attribute* to an *object*: $t = \mathbf{H}_{object} \odot \mathbf{H}_{attr}$
5: Get a noisy vector with a value: $\tilde{\mathbf{H}}_{val} = \mathbf{z}_{scene} \odot \mathbf{t}$
6: In $\mathbf{H}$ find a closest vector to a $\tilde{\mathbf{H}}_{val}$
7: Return it as *result*

To answer positional questions ("What objects are to the left of a given object?") we have to:

1. Query position of a given object $\mathbf{x}$;
2. Construct bundle $\mathbf{z}_{relate}$ of the shifted versions of vector $\mathbf{x}$;
3. Find the object in the item memory with similarity to $\mathbf{z}_{relate}$ greater than a threshold thr.

A detailed version of this algorithm is represented in Algorithm 3. The difference between $relate_{left}(\mathbf{z}_{scene}, object, thr)$ and $relate_{right}(\mathbf{z}_{scene}, object, thr)$ is in line 9 where $\mathbf{x}_{temp}$ is constructed as $\mathbf{x}_{temp} = \rho^i(\mathbf{H}_{x_{object}})$.

4.3. CLEVR example

Consider a scene depicted in Fig. 7 and the question *"There is a small gray block; are there any spheres to the left of it?"*.

As CLEVR question annotations contains programs from which they are generated the following steps are applied:

Algorithm 3: Get objects to the left of a given object.

procedure RELATE_LEFT($\mathbf{z}_{scene}, object, thr$)
  Get HD vector $\mathbf{H}_x$ for an attribute *x_coordinate* from $\mathbf{H}$
  Get HD vector $\mathbf{H}_{object}$ for an *object* from $\mathbf{H}$
  Assign *x_coordinate* to an *object*: $\mathbf{t} = \mathbf{H}_{object} \odot \mathbf{H}_x$
  Get a noisy vector of an x of an object: $\tilde{\mathbf{H}}_{x_{object}} = \mathbf{z}_{scene} \odot \mathbf{t}$
  In $\mathbf{H}$ find the closest vector to a $\tilde{\mathbf{H}}_{x_{object}}$. It is an x of an object – $\mathbf{H}_{x_{object}}$
  Create a list for shifted vectors *temp_coord*
  while $i \leq n$ **do** ▷ n - number of objects in the scene
    Shift $\mathbf{H}_{x_{object}}$ on i to the left: $\mathbf{x}_{temp} = \rho^{-i}(\mathbf{H}_{x_{object}})$
    Append $\mathbf{x}_{temp}$ to *temp_coord*
    $i+ = 1$
  Bundle all vectors in *temp_coord* and get $\mathbf{z}_{relate}$
  $\tilde{\mathbf{H}}_{object} = \mathbf{z}_{scene} \odot \mathbf{z}_{relate}$
  Query $\mathbf{H}$ with $\tilde{\mathbf{H}}_{object}$
  if $sim(\tilde{\mathbf{H}}_{object}, \mathbf{H}_{object_i}) > thr$ **then**
    Add $\mathbf{H}_{object_i}$ to the list of objects *result*
  Return *result*

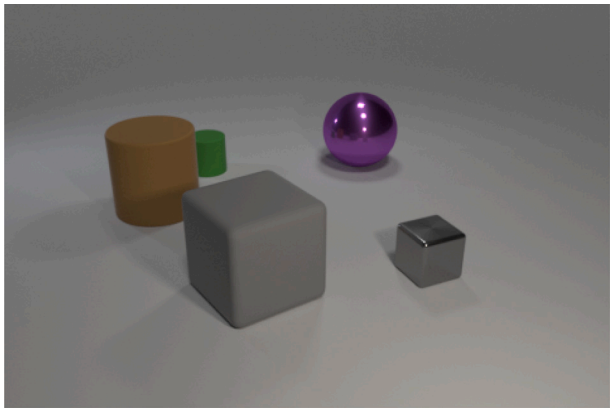


Fig. 7. Example image from the CLEVR validation split. The question is *"There is a small gray block; are there any spheres to the left of it?"*. The answer is *"Yes"*.

```
out = reas.filter_small(scene)
out = reas.filter_gray(scene, prev=out)
out = reas.filter_cube(scene, prev=out)
out = reas.unique(out)
out = reas.relate_left(out, scene)
out = reas.filter_sphere(scene, prev=out)
out = reas.exist(out)
```

Where *reas* is an instance of a VSM Reasoner Class, *filter_smth* methods that use *get_objects(SCENE, attribute, value, thr)* as a subroutine, and *unique*, *exist* are auxiliary methods.

4.4. Architecture

The overall model structure of the Vector Semiotic Model for VQA is represented in Fig. 8. The model consists of three parts: a scene parser, a question parser, and a VSM reasoner.

The scene parser uses Mask R-CNN (He, Gkioxari, Dollár, & Girshick, 2017) to extract object proposals for scene objects, then the proposals feed into a ResNet-34 attribute network that outputs the attributes of the detected objects and their 3D coordinates. The main assumption of the model is that there is a known number of object attributes n (*Color*, *Shape*, etc.) and the values of each attribute are from a discrete range $m_{attr_i}, i = 1...n$ (*Red*, *Blue*, *Square*, etc.), not necessarily the same for each attribute. Before processing any image–question pair, HD vectors for each attribute and its value are generated and stored in the item memory $\mathbf{H}$. Every HD vector in the item memory is stored with a corresponding label — name, which allows easy transferring from the symbolic to the vector representation and vice versa. The position vectors for each detected object $\mathbf{H}_x$, $\mathbf{H}_y$, $\mathbf{H}_z$ are also stored in the item memory. At the next stage, the results of the object proposal attribute classifications are fed into the SBWM module. This module stores a hierarchical representation of an abstract scene and considers every classified attribute-value pair as an event e_j in a causal matrix z_{object_i} of a corresponding object i. Then using the resultant semiotic representation of the objects and scenes, and extracting appropriate high-dimensional vectors from the item memory, the SBWM module constructs a vector representation $\mathbf{z}_{scene}$ of a scene causal matrix z_{scene}.

The question parser uses an attention-based seq2seq model (Bahdanau, Cho, & Bengio, 2015; Luong, Pham, & Manning, 2015) with LSTM (Hochreiter & Schmidhuber, 1997) cells. It translates an input question in the natural language into a sequence of procedures. Each procedure serves to answer a simple question like *"What is a color of a given object?"*, *"What objects are to the left of a given object?"*, etc., and perform HD operations on an input HD vector guided by a semiotic representation of a question. The sequence of procedures constitutes a program $\mathcal{P}$ that the question parser outputs.

The question parser consists of an encoder (which in turn consists of an embedding layer that converts the question to a sequence of vectors, and an LSTM module), a decoder (which consists of an embedding layer that converts the procedure tokens to vectors, an attention layer on encoder outputs (without learnable attention weight matrix), and also an LSTM module initialized by the hidden vectors of an encoder with concatenated respective vectors for different directions), and a classifier layer (which takes an output vector from a decoder and outputs the probabilities for each procedure token on a current step).

To reduce the number of the tokens required for program encoding, we decided to separate the prediction of the function and the argument by making a separate classification head for each of these. This helps to reduce the size of a dictionary of programs without a negative impact on the performance.

The parser's learning scheme is as follows: firstly, we pretrain the seq2seq part on a small set of question–program pairs in a supervised manner without using a reasoner; after that, we use the REINFORCE (Williams, 1992) algorithm to finetune the parser on question–answer pairs.

The VSM Reasoner receives a vector scene representation $\mathbf{z}_{scene}$ and a program $\mathcal{P}$. It sequentially executes procedures from $\mathcal{P}$ on $\mathbf{z}_{scene}$ in the manner that the output of a current procedure is fed together with $\mathbf{z}_{scene}$ into the next procedure. The output of the last procedure serves as the answer to the input question.

The REINFORCE part was done as follows: the output program is executed by NS-VQA or VSA executor, and the reward for the episode is binary – whether or not the answer generated by the program matches the ground truth answer; the reward is then averaged between all the episodes in a batch, and the baseline – a weighted mean of all previous rewards – is subtracted before calculating the loss function for variance reduction. The baseline is calculated as:

$$baseline = baseline \cdot reward\_decay + reward \cdot (1 - reward\_decay), \quad (5)$$

where the reward is calculated as a mean of binary rewards across the batch, and the reward decay is chosen as a means of stabilizing the baseline. The loss on each step is calculated as:

$$Loss_t = -\log \pi_\theta(s_t, a_t) \cdot baseline, \quad (6)$$

where π_θ is our current policy network (a question parser), s_t is the state – a decoder hidden state at the time step t, and a_t is the action – procedure sampled from the predicted distribution at the time step t.

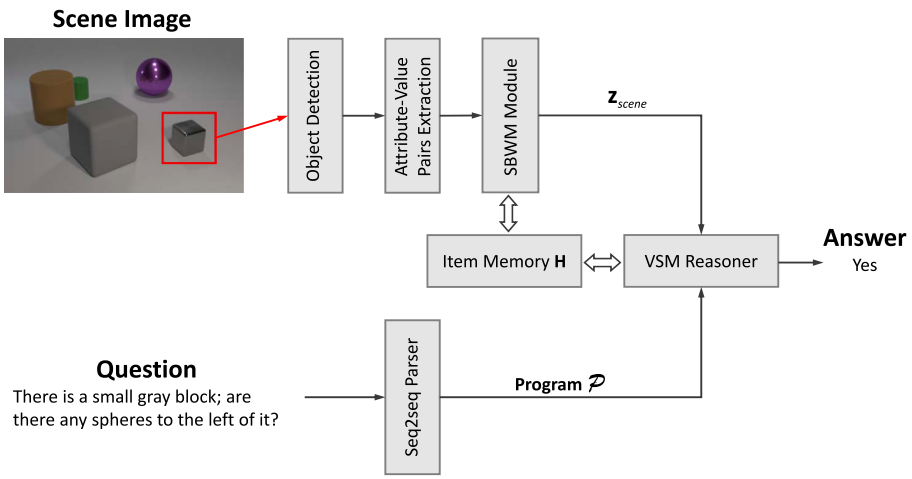


Fig. 8. Vector Semiotic Model for VQA. Before processing any image–question pair, high-dimensional vectors for each attribute and its value are generated and stored in the item memory **H**. The objects' regions detected in the input image are sent to classifiers to extract the attribute-value pairs. Based on the scene representation on the causal network the HD vector $\mathbf{z}_{scene}$ is built. At the same time, the input question is translated into a sequence of atomic VSA procedures (a program $\mathcal{P}$). Then program $\mathcal{P}$ is executed with an input vector $\mathbf{z}_{scene}$ and the result is sent to the output as an answer.

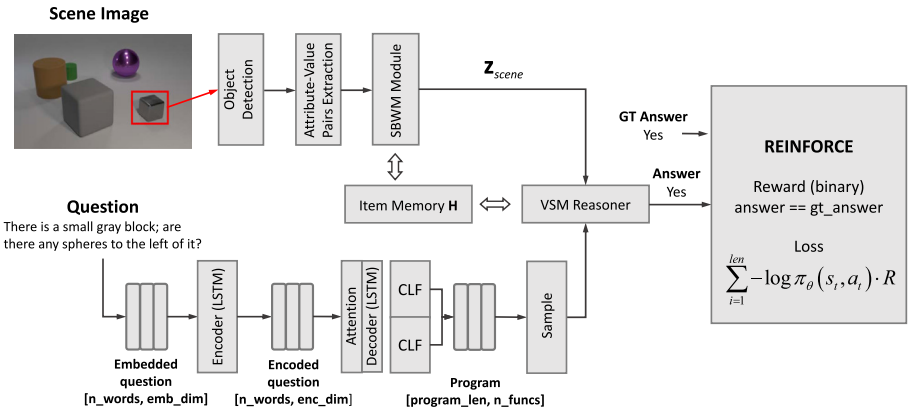


Fig. 9. Pipeline for the question parser training with REINFORCE. The embedded question is passed through an encoder, then the encoder outputs are sent to an attention module, and the decoder outputs are sent to a classifier (single or multi-head) that outputs the vectors of the probabilities for the procedure tokens. During the training, we sample the token from the categorical distribution based on these probabilities, and greedily choose the token with the highest probability during inference. These tokens are sent to a reasoner, the generated answer is compared to the ground truth answer, and the result is considered the reward for the episode — program. The reward is then used to compute the gradients for our policy, which consists of an entire question parser.

5. Experiments

5.1. Dataset

We demonstrate the performance of the proposed model on a diagnostic dataset CLEVR (Johnson, et al., 2017). CLEVR is a synthetic dataset that was proposed as a test-bed for VQA systems. Its main advantages are the simple visual scenes and complex long questions. The dataset is balanced that reduces any bias towards one or another possible answer. In the context of VQA, the bias means that the model can answer *"White"* to the question *"What color is the plate?"* even without looking at the image as there are mostly white plates in the dataset. An example of an image and the corresponding questions are shown in Fig. 10. It consists of 100,000 images and about a billion questions. The dataset is split into training (70,000 images, about 700,000 questions), validation (15,000 images, about 150,000 questions), and test (15,000 images, about 150,000 questions) parts. The answers are provided for all training and validation questions, thus performance is measured on a validation split. The questions in the natural language are generated from a functional program for each image and for each question a corresponding program is provided. Depending on the last functional block of a program, all questions are divided into five types: exist, count, compare integer, query attribute, and compare attribute. The examples for all types of questions are shown in Fig. 10.

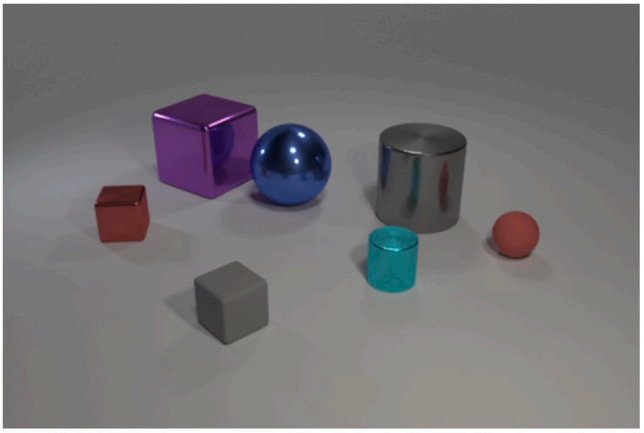
5.2. Training details

We inherit the learning scheme from the NS-VQA paper: first, we pretrain the question parser in a supervised manner using 270 question-program pairs (these were parsed from the CLEVR dataset and adapted to our function space). Then, we use REINFORCE for further training of the model (Fig. 9).

We encode the question words to vectors of size 300. The encoder recurrent module is a 2-layer bidirectional LSTM with the hidden size of 256. The procedure tokens are also encoded to vectors of size 300 and are passed through a unidirectional 2-layer LSTM with the hidden size of 512.

We used the Adam optimizer with learning rate $7 \cdot 10^{-4}$ for the pretraining part and 10^{-5} for the REINFORCE part.

We experimented with using different batch sizes for REINFORCE and discovered that the batch size 2 is sufficient for gradient estimation and yields the same final performance as the batch size 64 within roughly the same number of iterations, while significantly reducing the computational cost. The reward decay was chosen as 0.9.



Question (exist): Are there any cyan metallic cylinders to the right of the gray cylinder that is on the right side of the tiny red thing that is to the left of the big purple thing?

Question (count): How many other things are the same shape as the blue metal object?

Question (compare integer): Are there the same number of balls in front of the tiny cyan metallic cylinder and small gray rubber objects behind the tiny matte block?

Question (query attribute): What shape is the small red object right of the small gray cube?

Question (compare attribute): Do the small red object that is right of the large purple shiny object and the large object that is in front of the big sphere have the same shape?

Fig. 10. Example image and the corresponding questions from the CLEVR validation split. All types of questions are shown. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

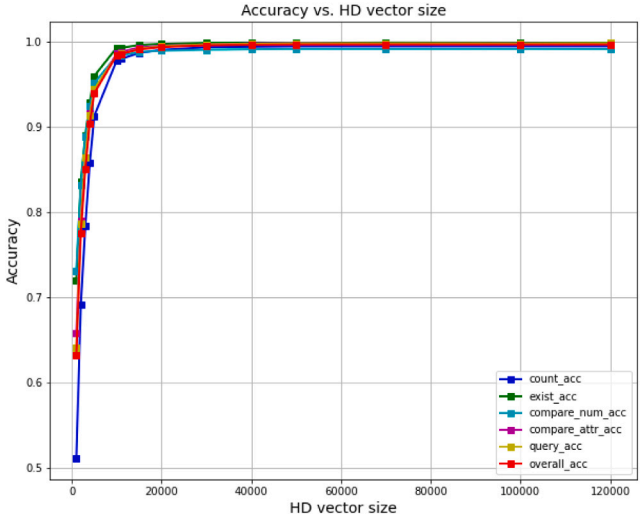


Fig. 11. Dependence of answer accuracy on the size of HD vectors.

5.3. Results

The main hyperparameter of any VSA system is the size of HD vectors. With an increase in the vector size on the one hand, we can store more vectors in one bundle, and on the other hand, we increase the accuracy of extracting vectors from the item memory **H**. In Fig. 11, we can see how accuracy changes depend on the vector size. As expected accuracy increases with an increase of the size, archives a maximum at 50,000 and then plateaus.

Another tuned hyperparameter is the threshold in the thresholded sum used for bundling. We evaluated the impact of the threshold for 15,000-dimensional vectors. The results are shown in Fig. 12. The increase in the threshold allows keeping more objects in a bundle, and thus more accurately extract them. It is seen from the plot that the best result achieved around the threshold 13, and a further increases do not affect the overall performance.

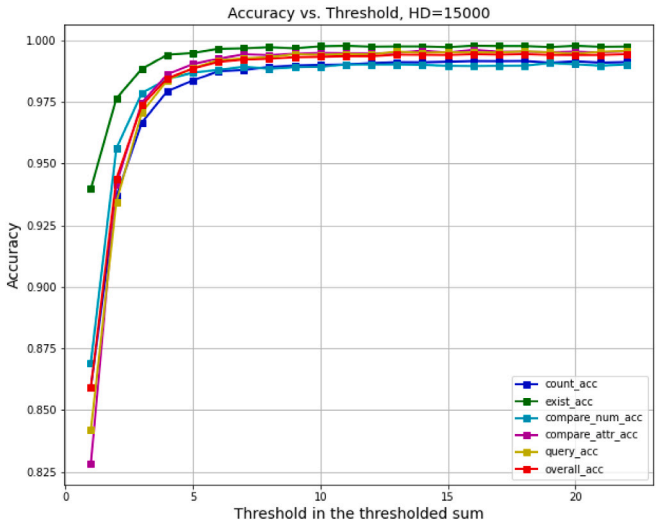


Fig. 12. Accuracy vs. Threshold in the thresholded sum.

Table 1
Accuracy metric is used. Results of fine-tuning the question parser with a VSM reasoner in the RL part. "Dim" stands for HD dimension. "CompN" and "CompA" stand for "Compare number"and "Compare attribute", respectively.

Dim	Count	Exist	CompN	CompA	Query	Overall
15k	91.2	94.2	95.4	95.5	94.1	94.0
30k	91.5	92.9	95.9	97.6	95.0	94.5

Table 2
Accuracy metric is used. Our model achieves comparable results to the state-of-the-art results. "MH" stands for multi-head version of the question parser. "CompN" and "CompA" stand for "Compare number" and "Compare attribute", respectively.

Model	Count	Exist	CompN	CompA	Query	Overall
Ours	99.2	99.8	98.9	99.6	99.5	99.4
MH	97.9	99.0	88.9	98.8	98.6	97.7
NS-VQA (Yi, et al., 2018)	**99.7**	**99.9**	**99.9**	**99.8**	**99.8**	**99.8**
NS-CL (Mao et al., 2019)	98.2	98.8	99.0	99.1	99.3	98.9
RN (Santoro, et al., 2017)	90.1	97.8	93.6	97.1	97.9	95.5
multiRN (Chang et al., 2018)	94.9	99.2	97.2	98.3	98.7	97.7

We fine-tuned the question parser (pretrained on 270 question–program pairs) with a VSM reasoner in the RL part and achieved results presented in Table 1. The random nature of VSA encoding increases variance and leads to difficulties in training. Changing the question parser to the parser trained without a VSM reasoner and using a VSM reasoner only on the evaluation phase leads to an accuracy increase (Table 2). We also demonstrate results for the multi-head (MH) version of the parser, while performance is slightly decreased it allows generalizing better for new combinations of attributes and values. Since NS-VQA (Yi, et al., 2018) provides results on a validation split we also evaluate our model on the validation split in Table 2. The results of other models (Chang et al., 2018; Mao et al., 2019; Santoro, et al., 2017) on a test split are added for a comparison.

6. Discussion

We have proposed a combination of a semiotic approach and VSA and apply it to VQA. The model combines neural computation, i.e., object detection and question parsing, with symbolic reasoning by VSA guided by SBWM representation.

In this work, we have used only part of the SBWM – an image component of a sign – that allows constructing a hierarchical representation of a scene and grounding scene objects in the agent's sensory inputs. The questions from CLEVR ask about external features, counting, or

relations. As the causal network on images encodes this information, it is necessary and sufficient for question answering. Other multi-modal tasks, such as Visual Dialog (Das, et al., 2017) and Visual Commonsense Reasoning (Zellers, Bisk, Farhadi, & Choi, 2018), require more elaborate reasoning and understanding of the context. In this case, the image component is no longer enough. Thus we have to use a significance component of a sign that represents the commonsense knowledge of an agent and a meaning component that stores the context of the dialog.

We use a multi-head classifier after the decoder network to predict the procedure and its arguments separately. That decreases the number of atomic procedures as the model predicts not*filter_red()* but *filter_color(red)* there *red* stands for any available color. It also simplifies the generalization to the new values of attributes compare to NS-VQA.

The proposed model processes images and questions separately, and all the information obtained from the image is considered in the vector description of the scene. On the one hand, this approach allows using of the same scene representation to answer different questions. On the other hand, it is ineffective for complex scenes. By complex scenes, we mean scenes with a large number of objects, which correspond to real-life images and not synthesized ones. Considering all objects, we encounter a limitation in the capacity of HD vectors. The vector bundles allow the storing of a finite number of vectors. Exceeding this number, we lose the ability to confidently extract vectors from the memory. Increasing the dimension of vectors, in theory, solves this problem (but we still cannot increase the dimension unboundedly). In practice, it also increases the computation time. To eliminate this problem, the model can process images and questions together and compose a description of the scene, guided by the information from the question. This can be achieved by using only the appropriate image components since the question tokens are treated as signs names.

Another limitation of the proposed approach for complex scenes is that it extracts objects based on annotated classes. Thus, scanty annotations lead to poor performance. As a solution, we can use an instance segmentation model trained on datasets with many categories (Gupta, Dollar, & Girshick, 2019).

7. Conclusion

In this paper, we have proposed the Vector Semiotic Model. It is a possible solution to the symbol grounding problem from a semiotic viewpoint. We applied the proposed solution to Visual Question Answering and demonstrated its performance on the CLEVR dataset. The obtained results are comparable to the state of the art.

We used the combination of the Sign-Based World Model cognitive architecture and Vector Symbolic Architectures. The Sign-Based World Model architecture allows representing scenes hierarchically and grounding symbols involved in reasoning in the agent's sensory input. We used Vector Symbolic Architectures as a computational and representational tool. Vector Symbolic Architectures allow preserving complex structures in a high-dimensional vector and extracting their parts via operations defined for these vectors. Thus, Vector Symbolic Architectures allow operating with these symbols as with numerical vectors. This combination of approaches allows bridging the gap between symbolic and neural (numerical) computations. The proposed solution might be applied to other tasks, such as a Visual Dialog (Das, et al., 2017), Visual Commonsense Reasoning (Zellers et al., 2018), and Loop Closure problem in simultaneous localization and mapping.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This work was supported by the Russian Science Foundation, project no. 20-71-10116.

Compliance with ethical standards

Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

References

Aitygulov, E., Kiselev, G., & Panov, A. I. (2018). Task and spatial planning by the cognitive agent with human-like knowledge representation. In A. Ronzhin, G. Rigoll, & R. Meshcheryakov (Eds.), *Interactive collaborative robotics* (pp. 1–12). Cham: Springer International Publishing.

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., et al. (2017). Bottom-up and top-down attention for image captioning and visual question answering. *ArXiv E-Prints*, arXiv:1707.07998, arXiv:1707.07998.

Andreas, J., Rohrbach, M., Darrell, T., & Klein, D. (2016). Neural module networks. In *2016 IEEE conference on computer vision and pattern recognition* (pp. 39–48). http://dx.doi.org/10.1109/CVPR.2016.12.

Bahdanau, D., Cho, K. H., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International conference on learning representations, ICLR 2015 - conference track proceedings* (pp. 1–15). arXiv:1409.0473.

Besold, T. R., & Kühnberger, K.-U. (2015). Towards integrated neural–symbolic systems for human-level AI: Two research programs helping to bridge the gaps. *Biologically Inspired Cognitive Architectures*, *14*, 97–110. http://dx.doi.org/10.1016/j.bica.2015.09.003, URL http://www.sciencedirect.com/science/article/pii/S2212683X15000468.

Bongini, P., Becattini, F., Bagdanov, A. D., & Bimbo, A. D. (2020). Visual question answering for cultural heritage. *IOP Conference Series: Materials Science and Engineering*, *949*(1), http://dx.doi.org/10.1088/1757-899X/949/1/012074, arXiv:2003.09853.

Butt, S., Ashraf, N., Siddiqui, M. H. F., Sidorov, G., & Gelbukh, A. (2021). Transformer-based extractive social media question answering on tweetqa. *Computación Y Sistemas*, *25*(1).

Chang, S., Yang, J., Park, S., & Kwak, N. (2018). Broadcasting convolutional network for visual relational reasoning. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Computer vision – ECCV 2018* (pp. 780–796). Cham: Springer International Publishing.

Chen, B., Vondrick, C., & Lipson, H. (2021). Visual behavior modelling for robotic theory of mind. *Scientific Reports*, *11*(1), http://dx.doi.org/10.1038/s41598-020-77918-x, cited By 0, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85099354828&doi=10.1038.

Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J. M., et al. (2017). Visual dialog. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Gayler, R. W. (1998a). Multiplicative binding, representation operators & analogy. In *Advances in analogy research* (pp. 1–4).

Gayler, R. W. (1998b). Multiplicative binding, representation operators, and analogy. In *Advances in analogy research: integr. oftheory and data from the cogn., comp., and neural sciences*. New Bulgarian University.

Gayler, R. W., & Wales, R. (1998). Connections, binding, unification and analogical promiscuity. In K. Holyoak, D. Gentner, & B. Kokinov (Eds.), *Advances in analogy research: Integration of theory and data from the cognitive, computational, and neural sciences* (pp. 181–190). New Bulgarian University, Sofia, URL http://cogprints.org/500/.

Gupta, A., Dollar, P., & Girshick, R. (2019). Lvis: A dataset for large vocabulary instance segmentation. *2019-June*, In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition* (pp. 5351–5359). arXiv:1908.03195.

Gupta, D., Suman, S., & Ekbal, A. (2021). Hierarchical deep multi-modal network for medical visual question answering. *Expert Systems with Applications*, *164*, Article 113993. http://dx.doi.org/10.1016/j.eswa.2020.113993, URL https://www.sciencedirect.com/science/article/pii/S0957417420307697.

Gurari, D., Li, Q., Lin, C., Zhao, Y., Guo, A., Stangl, A., et al. (2019). Vizwiz-priv: A dataset for recognizing the presence and purpose of private visual information in images taken by blind people. In *2019 IEEE/CVF conference on computer vision and pattern recognition* (pp. 939–948). http://dx.doi.org/10.1109/CVPR.2019.00103.

Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, *42*(1), 335–346. http://dx.doi.org/10.1016/0167-2789(90)90087-6, URL http://www.sciencedirect.com/science/article/pii/0167278990900876.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. B. (2017). Mask R-CNN. In *2017 IEEE international conference on computer vision* (pp. 2980–2988).

He, Y., Zhu, Z., Zhang, Y., Chen, Q., & Caverlee, J. (2020). Infusing disease knowledge into BERT for health question answering, medical inference and disease name recognition. In *Proceedings of the 2020 conference on empirical methods in natural language processing* (pp. 4604–4614). Online: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/2020.emnlp-main.372, URL https://www.aclweb.org/anthology/2020.emnlp-main.372.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*, 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., & Girshick, R. (2017). CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*.

Kanerva, P. (1997). Fully distributed representation. In *Real World Computing Symposium* (pp. 358–365).

Kanerva, P. (2009). Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, *1*(2), 139–159.

Kim, J.-H., Jun, J., & Zhang, B.-T. (2018). Bilinear attention networks. *ArXiv E-Prints*, arXiv:1805.07932, arXiv:1805.07932.

Kiselev, G., Kovalev, A., & Panov, A. I. (2018). Spatial reasoning and planning in sign-based world model. In S. O. Kuznetsov, G. S. Osipov, & V. L. Stefanuk (Eds.), *Artificial intelligence* (pp. 1–10). Cham: Springer International Publishing.

Kiselev, G. A., & Panov, A. I. (2017). Synthesis of the behavior plan for group of robots with sign based world model. In A. Ronzhin, G. Rigoll, & R. Meshcheryakov (Eds.), *Interactive collaborative robotics* (pp. 83–94). Cham: Springer International Publishing.

Kiselev, G., & Panov, A. (2019). Hierarchical psychologically inspired planning for human-robot interaction tasks. In A. Ronzhin, G. Rigoll, & R. Meshcheryakov (Eds.), *Interactive collaborative robotics* (pp. 150–160). Cham: Springer International Publishing.

Kleyko, D., Davies, M., Frady, E. P., Kanerva, P., Kent, S. J., Olshausen, B. A., et al. (2021). Vector symbolic architectures as a computing framework for nanoscale hardware. (pp. 1–28). arXiv:2106.05268, URL http://arxiv.org/abs/2106.05268.

Kleyko, D., Osipov, E., Gayler, R. W., Khan, A. I., & Dyer, A. G. (2015). Imitation of honey bees' concept learning processes using vector symbolic architectures. *Biologically Inspired Cognitive Architectures*, *14*, 57–72. http://dx.doi.org/10.1016/j.bica.2015.09.002.

Kovalev, A. K., & Panov, A. I. (2019). Mental actions and modelling of reasoning in semiotic approach to AGI. In P. Hammer, P. Agrawal, B. Goertzel, & M. Iklé (Eds.), *Artificial general intelligence* (pp. 121–131). Cham: Springer International Publishing.

Kovalev, A. K., Panov, A. I., & Osipov, E. (2020). Hyperdimensional representations in semiotic approach to AGI. In B. Goertzel, A. I. Panov, A. Potapov, & R. Yampolskiy (Eds.), *Artificial general intelligence* (pp. 231–241). Cham: Springer International Publishing.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images.

Laiho, M., Poikonen, J., Kanerva, P., & Lehtonen, E. (2015). High-dimensional computing with sparse vectors. In *2015 IEEE Biomedical Circuits and Systems Conference* (pp. 1–4).

Li, L. H., Yatskar, M., Yin, D., Hsieh, C.-J., & Chang, K.-W. (2019). VisualBERT: A simple and performant baseline for vision and language. In *Arxiv*.

Lobry, S., Marcos, D., Murray, J., & Tuia, D. (2020). RSVQA: Visual question answering for remote sensing data. *IEEE Transactions on Geoscience and Remote Sensing*, *58*(12), 8555–8566. http://dx.doi.org/10.1109/TGRS.2020.2988782.

Lu, J., Batra, D., Parikh, D., & Lee, S. (2019). ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*.

Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *ArXiv*, arXiv:1508.04025, URL https://re-work.co/blog/deep-learning-ilya-sutskever-google-openai, http://arxiv.org/abs/1508.04025.

Ma, L., Lu, Z., & Li, H. (2015). Learning to answer questions from image using convolutional neural network. arXiv:1506.00333.

Malinowski, M., Rohrbach, M., & Fritz, M. (2015). Ask your neurons: A neural-based approach to answering questions about images. In *2015 IEEE international conference on computer vision* (pp. 1–9). http://dx.doi.org/10.1109/ICCV.2015.9.

Manmadhan, S., & Kovoor, B. C. (2020). Visual question answering: a state-of-the-art review. In *Artificial intelligence review, Vol. 53* (8), (pp. 5705–5745). Springer Netherlands, http://dx.doi.org/10.1007/s10462-020-09832-7.

Manna, R., Das, D., & Gelbukh, A. (2021). Question-answering and recommendation system on cooking recipes. *Computación Y Sistemas*, *25*(1).

Mao, J., Gan, C., Kohli, P., Tenenbaum, J. B., & Wu, J. (2019). The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *ArXiv E-Prints*, arXiv:1904.12584, arXiv:1904.12584.

Montone, G., O'Regan, J., & Terekhov, A. V. (2017). Hyper-dimensional computing for a visual question-answering system that is trainable end-to-end. *ArXiv*, arXiv:1711.10185.

Osipov, G. S. (2015). Signs-based vs. Symbolic models. In G. Sidorov, & S. N. Galicia-Haro (Eds.), *Advances in artificial intelligence and soft computing* (pp. 3–11). Cham: Springer International Publishing.

Osipov, G. S., Panov, A. I., & Chudova, N. V. (2014). Behavior control as a function of consciousness. I. World model and goal setting. *Journal of Computer and Systems Sciences International*, *53*(4), 517–529. http://dx.doi.org/10.1134/S1064230714040121.

Panov, A. I. (2017). Behavior planning of intelligent agent with sign world model. *Biologically Inspired Cognitive Architectures*, *19*, 21–31. http://dx.doi.org/10.1016/j.bica.2016.12.001, URL http://www.sciencedirect.com/science/article/pii/S2212683X16300913.

Panov, A. I. (2019). Goal setting and behavior planning for cognitive agents. *Scientific and Technical Information Processing*, *46*(6), 404–415. http://dx.doi.org/10.3103/S0147688219060066.

Pathak, A., Manna, R., Partha Pakray, D. D., Gelbukh, A., & Bandyopadhyay, S. (2021). Scientific text entailment and a textual-entailment-based framework for cooking domain question answering. *Sādhanā*, *46*(24), http://dx.doi.org/10.1007/s12046-021-01557-9.

Plate, T. A. (2003). *Holographic reduced representations: Distributed representation for cognitive structures*. Stanford: Center for the Study of Language and Information (CSLI), USA.

Rachkovskij, D. A. (2001). Representation and processing of structures with binary sparse distributed codes. *IEEE Transactions on Knowledge and Data Engineering*, *3*(2), 261–276.

Roy, D. (2005). Semiotic schemas: A framework for grounding language in action and perception. *Artificial Intelligence*, *167*(1–2), 170–205. http://dx.doi.org/10.1016/j.artint.2005.04.007, URL http://linkinghub.elsevier.com/retrieve/pii/S0004370205001037.

Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., et al. (2017). A simple neural network module for relational reasoning. In *Advances in neural information processing systems*.

Schlegel, K., Neubert, P., & Protzel, P. (2020). A comparison of vector symbolic architectures. arXiv:2001.11797, URL http://arxiv.org/abs/2001.11797.

Schmidtke, H. R. (2018). Logical lateration – A cognitive systems experiment towards a new approach to the grounding problem. *Cognitive Systems Research*, *52*, 896–908. http://dx.doi.org/10.1016/j.cogsys.2018.09.008, URL https://www.sciencedirect.com/science/article/pii/S1389041717302760.

Schmidtke, H. (2021a). Multi-modal actuation with the activation bit vector machine. *Cognitive Systems Research*, *66*, 162–175. http://dx.doi.org/10.1016/j.cogsys.2020.10.022, URL https://www.sciencedirect.com/science/article/pii/S1389041720300966.

Schmidtke, H. R. (2021b). Reasoning and learning with context logic. *Journal of Reliable Intelligent Environments*, http://dx.doi.org/10.1007/s40860-020-00121-2.

Sheppard, E., & Lohan, K. S. (2020). Multimodal representation learning for human robot interaction. In *HRI '20, Companion of the 2020 ACM/IEEE international conference on human-robot interaction* (pp. 445–446). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3371382.3378265.

Singh, H., & Shekhar, S. (2020). STL-CQA: Structure-based transformers with localization and encoding for chart question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing* (pp. 3275–3284). Online: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/2020.emnlp-main.264, URL https://www.aclweb.org/anthology/2020.emnlp-main.264.

Steels, L. (2008). The symbol grounding problem has been solved. So what's next. In M. de Vega, A. Glenberg, & A. Graesser (Eds.), *Symbols and embodiment: Debates on meaning and cognition* (pp. 223–244). Oxford University Press.

Su, D., Xu, Y., Yu, T., Siddique, F. B., Barezi, E., & Fung, P. (2020). CAiRE-COVID: A question answering and query-focused multi-document summarization system for COVID-19 scholarly information management. In *Proceedings of the 1st workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*. Online: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/2020.nlpcovid19-2.14, URL https://www.aclweb.org/anthology/2020.nlpcovid19-2.14.

Su, W., Zhu, X., Cao, Y., Li, B., Lu, L., Wei, F., et al. (2020). VL-BERT: Pre-training of generic visual-linguistic representations. In *International conference on learning representations*. URL https://openreview.net/forum?id=SygXPaEYvH.

Talbot, B., Dayoub, F., Corke, P., & Wyeth, G. (2020). Robot navigation in unseen spaces using an abstract map. *IEEE Transactions on Cognitive and Developmental Systems*, 1. http://dx.doi.org/10.1109/TCDS.2020.2993855.

Tan, H., & Bansal, M. (2019). LXMERT: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 conference on empirical methods in natural language processing*.

Trifonas, P. P. (Ed.), (2015). *International handbook of semiotics* (p. 1325). Dordrecht: Springer Netherlands, http://dx.doi.org/10.1007/978-94-017-9404-6, URL http://link.springer.com/10.1007/978-94-017-9404-6.

Vo, H. Q., Phung, T. H., & Ly, N. Q. (2020). VQASTO: Visual question answering system for action surveillance based on task ontology. In *2020 7th NAFOSTED Conference on Information and Computer Science* (pp. 273–279). http://dx.doi.org/10.1109/NICS51282.2020.9335891.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine learning* (pp. 229–256).

Wu, Q., Shen, C., Wang, P., Dick, A., & v. d. Hengel, A. (2018). Image captioning and visual question answering based on attributes and external knowledge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(6), 1367–1381. http://dx.doi.org/10.1109/TPAMI.2017.2708709.

Wu, Q., Teney, D., Wang, P., Shen, C., Dick, A., & van den Hengel, A. (2017). Visual question answering: A survey of methods and datasets. *Computer Vision and Image Understanding, 163*, 21–40. http://dx.doi.org/10.1016/j.cviu.2017.05.001, Language in Vision, URL https://www.sciencedirect.com/science/article/pii/S1077314217300772.

Wu, Q., Wang, P., Shen, C., Dick, A., & Van Den Hengel, A. (2016). Ask me anything: Free-form visual question answering based on knowledge from external sources. In *2016 IEEE conference on computer vision and pattern recognition* (pp. 4622–4630). http://dx.doi.org/10.1109/CVPR.2016.500.

Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., & Tenenbaum, J. B. (2018). Neural-symbolic VQA: Disentangling reasoning from vision and language understanding. *ArXiv E-Prints*, arXiv:1810.02338, arXiv:1810.02338.

Yilmaz, O. (2015). Analogy making and logical inference on images using cellular automata based hyperdimensional computing. In *COCO'15*, (pp. 19–27). Aachen, DEU: CEUR-WS.org.

Yu, W., Wu, L., Deng, Y., Mahindru, R., Zeng, Q., Guven, S., et al. (2020). A technical question answering system with transfer learning. In *Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations* (pp. 92–99). Online: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/2020.emnlp-demos.13, URL https://www.aclweb.org/anthology/2020.emnlp-demos.13.

Yu, Z., Yu, J., Cui, Y., Tao, D., & Tian, Q. (2019). Deep modular co-attention networks for visual question answering. In *2019 IEEE/CVF conference on computer vision and pattern recognition* (pp. 6274–6283).

Yu, J., Zhu, Z., Wang, Y., Zhang, W., Hu, Y., & Tan, J. (2020). Cross-modal knowledge reasoning for knowledge-based visual question answering. *Pattern Recognition, 108*, Article 107563. http://dx.doi.org/10.1016/j.patcog.2020.107563, URL https://www.sciencedirect.com/science/article/pii/S0031320320303666.

Zellers, R., Bisk, Y., Farhadi, A., & Choi, Y. (2018). From recognition to cognition: Visual commonsense reasoning. *CoRR*, arXiv:1811.10830, arXiv:1811.10830, URL http://arxiv.org/abs/1811.10830.

Zhang, W., Deng, Y., Ma, J., & Lam, W. (2020). Answerfact: Fact checking in product question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing* (pp. 2407–2417). Online: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/2020.emnlp-main.188, URL https://www.aclweb.org/anthology/2020.emnlp-main.188.

Zhou, B., Tian, Y., Sukhbaatar, S., Szlam, A., & Fergus, R. (2015). Simple baseline for visual question answering. arXiv:1512.02167.