Compositional Generalization and Neuro-Symbolic Architectures

Savitha Sam Abraham*, **Marjan Alirezaie***

Centre for Applied Autonomous Sensor Systems (AASS), Örebro University, Sweden
savitha.sam-abraham@oru.se, marjan.alirezaie@oru.se

Abstract

Compositional generalization is the ability to understand novel combinations of known concepts. Although it is considered as an innate skill for humans, recent studies have shown that neural networks lack this characteristic. In this paper, we focus on compositional generalization with respect to the two specific tasks of word problem solving and visual relation recognition and propose a neuro-symbolic solution, using DeepProbLog, that addresses the problem of compositionality in state-of-the-art neural systems for these tasks.

Introduction

Compositional generalization is an instinctive ability of humans to decipher novel combinations of primitive concepts, once they are familiar with the primitive concepts and have observed a few instances of the interactions between these concepts. For example, if a person knows the meaning of 'jump', 'run' and 'run twice', she can immediately understand the meaning of 'jump twice'. Similarly, if a person can identify a cube, sphere, a red sphere and a metallic sphere, it is straightforward for her to recognize a red metallic cube even though she has not seen one before. Recent research (Lake and Baroni 2018), (Klinger et al. 2020) has demonstrated through extensive experimental studies that purely neural models lack the characteristic of compositional generalization. While (Lake and Baroni 2018) showed that sequence to sequence models were not effective in learning compositional rules for generalization in the domain of natural language processing (NLP), (Klinger et al. 2020) demonstrated how neural models failed to generalize compositionally in relation recognition in the domain of computer vision.

Word problems or story problems in the mathematics domain are short narratives describing a real-world problem scenario in natural language. The narrative is accompanied by a question about the problem scenario. Answering this question requires one to first extract pertinent numerical facts from the problem, identify the unknown quantity, form an equation and finally solve it. The following are examples of two word problems, mapped to their equations. These are single-equation single-operation/step problems as they can be solved with a single equation that involves a single operation (addition in Problem 1, subtraction in Problem 2).

Problem 1: John had 10 apples. He bought 11 more. How many apples does he have now?
Equation: $x = 10 + 11$
Solution: $x = 21$

Problem 2: John had 10 apples. He ate 2. How many apples does he have now?
Equation: $x = 10 - 2$
Solution: $x = 8$

Compositional generalization in the context of word problem solving is the ability to solve new problems involving equations with multiple operations having observed only single-equation single-operation word problems. Consider the following problem:

Problem 3: John had 10 apples. He bought 11 more. He ate 2. How many apples does he have now?
Equation: $x = 10 + 11 - 2$
Solution: $x = 19$

Although the above problem is a new type of problem as the equation it maps to involves multiple operations (unlike Problems 1 and 2), it is composed of sentences that are similar to the ones in Problems 1 and 2. If a student knows how to solve Problems 1 and 2, solving Problem 3 is straightforward. But, as we will show in this paper, the state-of-the-art word problem solvers fail to solve a multi-operation problem like Problem 3 at test time, if these are just trained on single-operation problems like 1 and 2. That is, these solvers fail to generalize compositionally. In order to evaluate the ability of automatic word problem solvers to generalize compositionally, we create a dataset by collecting (from existing datasets) a specific type of word problems called *Change Problems* (Powell and Fuchs 2018). Change problems involve scenarios where there is an entity of interest with an initial value, that gets updated (either increased or decreased) to a new value as a result of an action. Problems 1, 2 (single update change) and 3 (two update change) are all examples of change problems.

Compositionality is also the focus of recent research in computer vision due to multi-modality in visual features.

*These authors contributed equally.

Figure 1: Considering the order of images, the *True* label in (1) indicates either a red triangle or a blue circle or an intersection between them is of importance. *True* label in (2) discards the color constraints. *False* label in (3) confirms that intersection with a triangle results in label *True*. So, one can infer (4) as *True*, regardless of the new combination of colors for the shapes.

Figure 1 shows an example of visual-spatial intelligence tests, rather easy for humans to address, but problematic for neural-based methods due to compositionality in the color/position/shape of objects. Although existing deep learning methods have shown promising results in visual relation recognition, they tend to fail with compositional generalization (Klinger et al. 2020). This is largely due to distribution-shift where a model is tested with new combinations of features never observed during training. In this paper, we study compositionality w.r.t color, shape and position of the objects in the image (as shown in Figure 1). In order to evaluate compositionality in visual relation recognition, we have devised our own dataset, rather than using available benchmarks (Johnson et al. 2016; Santoro et al. 2017). One reason behind such decision is that the aforementioned benchmarks are associated with language input which requires a parallel line of processing separated from the visual recognition task. Furthermore, by generating our own dataset, we are in control of generating different types of compositionality related to color, shape, position and number of objects.

One can address inability of neural models to generalize compositionally by providing it with more data that has sufficient number of instances of every combination of concepts we are interested in. There are tasks where providing more data may not be practical, but it may be instead more convenient and efficient to provide knowledge about the tasks that can aid the models in generalizing better. We, in this paper, show that incorporating task specific knowledge into purely neural models help address its inability to generalize compositionally. For this, we propose a neuro-symbolic approach using DeepProbLog (Manhaeve et al. 2018) for both word problem solving and visual (spatial) relation recognition, and, show how by modelling task specific knowledge in logic, such models can improve the performance of neural models w.r.t compositional generalization.

Our contributions are the following - for the two tasks of word problem solving and visual relation recognition, we

1. create datasets to test compositional generalization.
2. analyze compositional generalization ability of state-of-the-art neural models using the above datasets.
3. develop neuro-symbolic models based on DeepProbLog to improve compositional generalization ability through inclusion of task specific knowledge.

Related Work

In this section we briefly survey the following areas: automatic word problem solving, visual relation recognition, and finally, neuro-symbolic modelling.

Automatic Math Word Problem Solving

There has been a surge of research in automatic math word problem solving in the past six years. (Zhang et al. 2019) gives an extensive survey on automatic word problem solvers. While the earlier systems used a purely symbolic approach to solving, the more recent ones are data-driven based systems. Some consider it as a structure prediction problem, where the structure to be predicted is the expression tree corresponding to the equation that the word problem is mapped onto ((Roy and Roth 2016), (Kushman et al. 2014)). Since these required manual feature engineering, the community has transitioned to deep learning approaches. Most of the existing neural models for math word problem solving utilize Seq2Seq models (Luong, Kayser, and Manning 2015), considering the source sequence as the word problem and the target sequence as the equation. More recently (Zhang et al. 2020) and (Xie and Sun 2019) introduced graph-based encoders to encode the graph-based representation of the word problem and tree-based decoders to generate expression trees for the word problem.

The research on math word problem solvers started off with datasets like, AddSub (Kushman et al. 2014) and SingleOp (Roy, Vieira, and Roth 2015), containing simple single-equation word problems. Although some now consider this as a solved research problem, (Patel, Bhattamishra, and Goyal 2021) illustrated through experiments that the existing state-of-the-art solvers relied on superficial heuristics to map the word problems to equations. They created a challenge dataset named SVAMP, that contained problems created by slightly varying the problems from existing datasets like AddSub and SingleOp. These variations were created by adding relevant or irrelevant information to the problem or changing the order of phrases in the word problem. It was shown that the state-of-the-art solvers trained on existing datasets showed poor performance when tested on SVAMP dataset. In this paper, we introduce another challenge - that of compositional generalization and show that while state-of-the-art neural solvers fail to generalize compositionally, the proposed DeepProbLog solver exhibits promising results. To test the ability of solvers to generalize compositionally, we train both the state-of-the-art solvers and our DeepProbLog-based model on simple single-step, single-equation *change* problems from AddSub and SingleOp (like, Problem 1 and Problem 2) and then test the trained models on multi-step, single-equation problems (like Problem 3).

Visual Relation Recognition

In recent years, visual relation recognition has also been at the centre of deep learning research in computer vision under different names such as visual question answering (VQA). Most of the deep learning approaches for VQA are based on convolutional neural networks (CNNs) together with relation networks (RNs) (Santoro et al. 2017). For instance, a two-stage Relation Network (2S-RN) (Messina

et al. 2018) has achieved 95% accuracy in answering visual relational questions in CLEVR dataset (Johnson et al. 2016). (Ding et al. 2021) also proposed a neural object-based attention model for solving spatio-temporal reasoning task and was tested on CLEVRER (Yi et al. 2020) and CATER (Girdhar and Ramanan 2019) datasets. The RN-augmented architecture (Santoro et al. 2017) has outperformed the state-of-the-art in tackling relational inferences about complex scenes in CLEVR. Moreover, (Dai, Zhang, and Lin 2017) proposed an integrated framework called Deep Relational Network (DR-Net), that could exploit statistical dependencies between objects, their spatial configurations and appearances. DR-Net is composed of several modules such as RCNN-based object detection (He et al. 2017), pair filtering, and joint recognition used for exploring all the possible spatial relations between any two pairs of objects. A large number of contributions in visual relation recognition has been evaluated on the datasets involving both visual scenes and text annotations, i.e., the proposed models include modules to learn features from the language input in parallel.

Towards more general purpose architectures for relation recognition, (Shanahan et al. 2020) proposed PrediNet, an end-to-end network inspired by symbolic AI based on first-order predicate calculus, that learns and maps vectorial representations from raw pixels to propositions with explicit relational structure. PrediNet (upon a customized dataset) has shown its capacity to address compositionality to a degree, thanks to the core of the network that allows for parallel processing of information organized into small chunks that contribute to learning reusable representations. However, most proposed deep relation recognition methods perform poorly w.r.t compositional generalization (Klinger et al. 2020). We show in this paper that a neuro-symbolic model for relation recognition can offer solutions to the problem of compositionality observed in deep neural models.

Neuro-symbolic Models

The area of Neuro-Symbolic Artificial Intelligence (NeSy AI) focuses on integrating the neural and symbolic AI in such a way that the desirable characteristics of both are inherited efficiently and effectively (De Raedt et al. 2020). The research in the area has resulted in the development of multiple neuro-symbolic frameworks like DeepProbLog (Manhaeve et al. 2018), Concept Learner (Mao et al. 2019), Neural-Grammar-Symbolic Model (NGS) (Li et al. 2020), Neuro Symbolic Forward Reasoner (NSFR) (Shindo, Dhami, and Kersting 2021). These frameworks differ in the way the symbolic part is made differentiable and incorporated into the training of the neural network part of the framework. In this paper, we use DeepProbLog to implement neuro-symbolic solutions for the two tasks. DeepProbLog integrates neural networks with the very expressive ProbLog paradigm (De Raedt, Kimmig, and Toivonen 2007), and allows explicit representation of task-specific knowledge in ProbLog (unlike Concept Learner, which is based on quasi-symbolic program execution) and is based on a gradient-based learning algorithm (unlike NGS - in which the module that propagates loss back to the neural network is task-dependent and has to be redefined for every task).

DeepProbLog for Compositional Generalization

As mentioned earlier, DeepProbLog combines neural networks with the probabilistic programming language, ProbLog (De Raedt, Kimmig, and Toivonen 2007), by extending ProbLog with neural predicates. In Problog, a fact f can be associated with a probability p, as $p :: f$, indicating f is true with probability p and false with probability $1 - p$. An annotated disjunction in Problog is represented as:

$$p1 :: h1; ...; p_n :: h_n \text{ :- } b_1, ..., b_m$$

where, $\Sigma p_i <= 1$. The above translates to 'if all b_i hold, then either one of the h_j is true with probability p_j, or none is true with probability $1-\Sigma p_i$'. DeepProbLog, through neural predicates, enables us to use neural networks to provide probabilities associated with a fact. A neural annotated disjunction is of the following form:

$$nn(m_r, [X_1, ..., X_k], O, [y_1, ..., y_n]) :: r(X_1, ..., X_k, O)$$

where m_r represents a neural network that generates a probability distribution $p_{m_r}(O|X = x)$ over the domain $[y_1, ..., y_n]$, given a specific input $x = [x_1, ..., x_k]$ (raw data - e.g., sequence of words, tensor representing an image). The above translates to '$r(X_1, ..., X_k, y_i)$ is true with probability $p_{m_r}(O = y_i|X_1 = x_1, ..., X_k = x_k)$'. The neural network is a discriminative classifier, that may use an encoder of choice (e.g., convolutional, recurrent neural networks), but the output layer is expected to normalize the inputs it receives into a probability distribution (like softmax layer). Thus, a DeepProbLog program is comprised of a set of neural annotated disjunctions and neural facts in addition to a set of probabilistic facts, and rules as in ProbLog program. The weights of the neural networks represented by the neural predicates in a DeepProbLog program is learned using a gradient-based learning approach. DeepProbLog relies on gradient semiring in Algebraic ProbLog (Kimmig, Van den Broeck, and De Raedt 2011) to compute the gradient of the loss with respect to the outputs of the neural networks (which are the probabilities associated with the neural facts). The gradient of the neural network outputs with respect to the weights of the neural network is computed as in the standard backpropagation algorithm (using chain rule).

We will next describe the details of the proposed DeepProbLog-based word problem solver and visual relation recognizer, and the solving pipeline as depicted in the grey box in Figures 3 and 4.

DeepProbLog for Word Problem Solving

We break down the task of word problem solving into two components - the first is that of perception or understanding and extracting relevant facts from the natural language description and the second that of performing mathematical reasoning on these facts.

Perception In this section, we describe the neural networks associated with the two neural predicates in the program. In order to address the problem of compositionality, we process the word problem sentence-wise, representing each sentence in logical form. We use a representation simi-

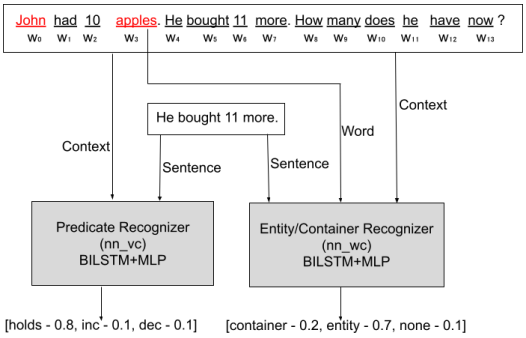

Figure 2: Processing the second sentence '*He bought 11 more*' - Figure shows the neural networks represented by the two neural predicates, nn_vc and nn_wc, that extracts the predicate and the container, entity mentions in the sentence.

lar to the one described in (Kushman et al. 2014). The set of predicates used are shown below:

1. *holds(C, E, N)*: represents the fact that a container 'C' holds 'N' number of the entity 'E'.
2. *increase(C, E, N_i)*: represents the fact that the number of items of entity 'E' held by the container 'C' is increased by an amount, 'N_i'.
3. *decrease(C, E, N_d)*: represents the fact that the number of items of entity 'E' held by the container 'C' is decreased by an amount, 'N_d'.

The logical representation for Problem 1 would be:

- 'John had 10 apples'
  holds(John, apples, 10)
- 'He bought 11 more'
  increase(John, apples, 11)
- 'How many does he have now?'
  holds(John, apples, ?X)

Hence, given a sentence, the goals of perception are to identify the predicate associated with the sentence (*holds* or *increase* or *decrease*) and the arguments associated with this predicate (the container and the entity). Figure 2 shows the two neural networks, both discriminative classifiers, involved - *predicate recognizer* and *entity/container recognizer*. A predicate recognizer takes the specific sentence - S, and its context, which is the entire word problem - P, as its two inputs and generates a probability distribution over the domain {holds, increase, decrease}. The context is in some cases important to decide whether the verb in the sentence represents an action that results in an increase or decrease of the quantity of interest. Consider the following example:

> **Problem 4**: There were 6 roses in the vase. Mary cut some roses from her flower garden. There are now 16 roses in the vase. How many did she cut?
> **Equation**: $6 + x = 16$
> **Solution**: $x = 10$

The predicate associated with the second sentence is 'increase' as it results in an increase in the number of roses in the vase. If the sentence is considered in isolation, there are possibilities of associating it with 'decrease', as it is describing an action ('cut') that caused a reduction in the number of roses in the flower garden. The neural network uses a Bidirectional Long Short Term Memory (BiLSTM) to generate contextual representations of the sentence and the word problem, which are then consumed by a MultiLayer Perceptron (MLP) to generate the required probability distribution.

A container/entity recognizer takes three inputs - the sentence - S, the word problem - P and the index - W_i (some number between 0 and 13 in case of Problem 1 as shown in Figure 2) of a word in the word problem. The network predicts the probability that the corresponding word is a container or entity or neither for the specific sentence. The sentence cannot be processed in isolation, as it might not have a container or entity mentioned in it directly. For example, in the sentence, '*He bought 11 more*', in Figure 2, the pronoun 'he' is used to refer to the actual container 'John' which is in the previous sentence. Similarly, the entity, 'apples' is also absent in the sentence. As in the case of predicate recognizer, we use a BiLSTM to generate contextual representations of the sentence and the word problem. The representation of the specific word, W_i, is obtained from the word problem's representation which is then concatenated with the sentence before passing it on to MLP. MLP generates a probability distribution, over the domain {container, entity, none}, for the word. We invoke the network once for each word in the word problem - since containers and entities are nouns, we invoke the network only for words that are nouns in the word problem. Thus, the recognizer gives a probability distribution for each noun in the word problem.

Reasoning Figure 3 shows the different phases in solving, starting from perception. The knowledge provided enables it to derive the equation associated with a word problem. In the case of *change* problems, the query is regarding an unknown initial value or update or final value.

The query predicate posed to the program is `?wps(Tok_Sentences, POS_Sentences, Quantities, Solution)`, where `Tok_Sentences` is the list of sentences/constituents in the word problem [1] - each constituent represented as a list of words/tokens in it, `POS_Sentences` is the list of part of speech tags of words in the problem[2], `Quantities` is the list of numerical quantities mentioned in the word problem and finally, `Solution` - represents the solution of the word problem. While training, the actual solution is provided in `Solution` and the goal is to find a proof for the posed query. While testing, `Solution` will be assigned the solution predicted by the program.

The perception phase, as mentioned earlier, employs neural networks for predicate recognition and entity/container recognition. These networks are represented by the neural predicates, `nn_vc` (vc for verb categorizer as the verb in the sentence determines the predicate - holds/increase/decrease) taking as inputs the `Sentence` and `Context`

[1] Constituents are obtained using Stanford constituency parser (Zhu et al. 2013)

[2] POS tags are obtained using Stanford parser (Klein and Manning 2003)

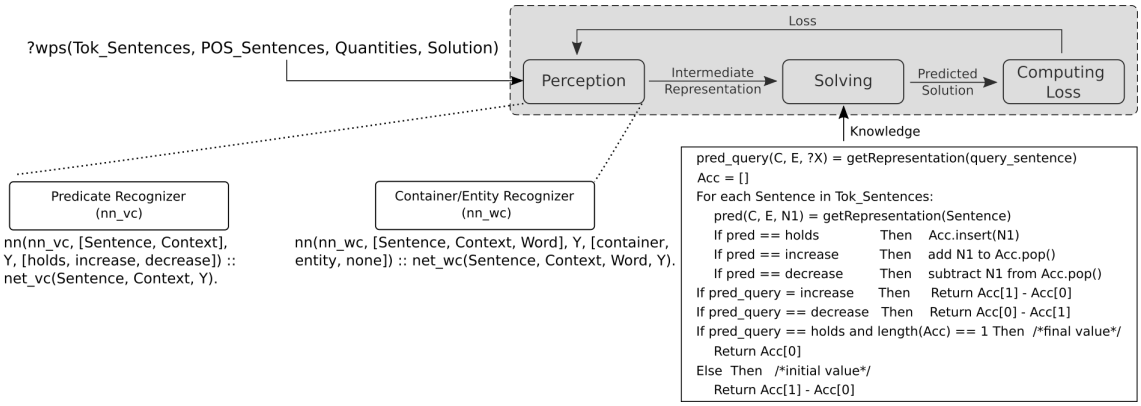

Figure 3: The DeepProbLog pipeline in solving word problems.

and predicting `Y` over the domain `[holds, increase, decrease]` and `nn_wc` (wc for word categorizer identifying whether the word is a container/entity/none) taking as inputs `Sentence`, `Context` and a specific `Word` and predicting `Y` over the domain `[container, entity, none]`.

The knowledge provided to the solver guides it to arrive at an equation and a solution. *We assume that the temporal order of events in the word problem is same as the order in which it is narrated*. This allows sequential processing of the sentences in the word problem. The pseudocode of the knowledge provided is shown in Figure 3. *getRepresentation(Sentence)* invokes the neural networks responsible for perception, through the neural predicates *nn_vc* and *nn_wc*, to obtain the intermediate representation of a sentence. The solving process starts from the last sentence in the word problem, which is assumed to be the sentence posing the query (*query_sentence* in pseudocode). The solver first gets the representation of the query sentence (*holds(John, apples, ?X)* in case of Problem 1). This gives information about the predicate associated with the query sentence (*pred_query* in pseudocode) and the container and entity of interest. If the predicate is holds, then the unknown value is either the initial/final value (final value is queried for in case of Problem 1). If the predicate is increase/decrease, then the unknown value queried for is the change or the amount by which the value got updated. The *for loop* in the pseudocode indicates that the solver processes each sentence in the word problem, starting from the first sentence - getting a representation for each sentence (a sentence is relevant if it is about the container and entity mentioned in the query). Based on the predicate associated with each sentence (*pred* = holds/increase/decrease), the numerical quantity associated with the sentence is appended into an accumulator (*Acc* in pseudocode) directly or the value in the accumulator is increased or decreased by the quantity. The processing for Problem 1 is shown below:

Sentence 1: *holds(John, apples, 10)* $->$ Acc = [10]
Sentence 2: *increase(John, apples, 11)* $->$ Acc = [10+11]
Sentence 3 (Query): *holds(John, apples, ?X)* $->$ Result = Acc[0] = 21

DeepProbLog for Relation Recognition

To show the role of neuro-symbolic frameworks in addressing compositional generalization in vision, we have defined a relation recognition task as the task of *identifying images that involve at least a triangle intersecting with any other geometrical shape*. As in the case of word problem solving, we break down the task of visual relation recognition into two parts - perception and reasoning.

Perception

Given that the state-of-the-art deep learning methods in (multi) object detection, like MASK R-CNN (He et al. 2017), show good performance, we have simplified visual relation recognition into a classification task that accepts as input visual scenes together with the bounding boxes of objects in the scene provided by a pre-trained object detection model like MASK R-CNN. Since the type of shapes of the objects in the scene is important, we dedicated the perception part of our DeepProbLog model to shape recognition (see Figure 1). More specifically, we have implemented a CNN-MLP neural model to classify input images into 3 classes - *circle*, *triangle* and *rectangle*.

Reasoning

Figure 4 illustrates the same solving pipeline depicted in Figure 3, however modified for relation recognition.

The query predicate is `?scene(Images, BBoxes, Solution)`, where `Images` is the list of images, one for each object in the given input image, `BBoxes` is the list of bounding boxes of the objects and `Solution` represents the actual (during training) or the predicted class label (during testing) which is either True/False.

Given the input data, the perception phase employs a neural network for shape recognition. The network is represented by the neural predicate, `nn_shape`, taking as input one element of `Images` (an object) and predicting `Y` over the domain `[circle, triangle, rectangle]`.

The knowledge associated with the solving process identifies the spatial relations between each pair of shapes and

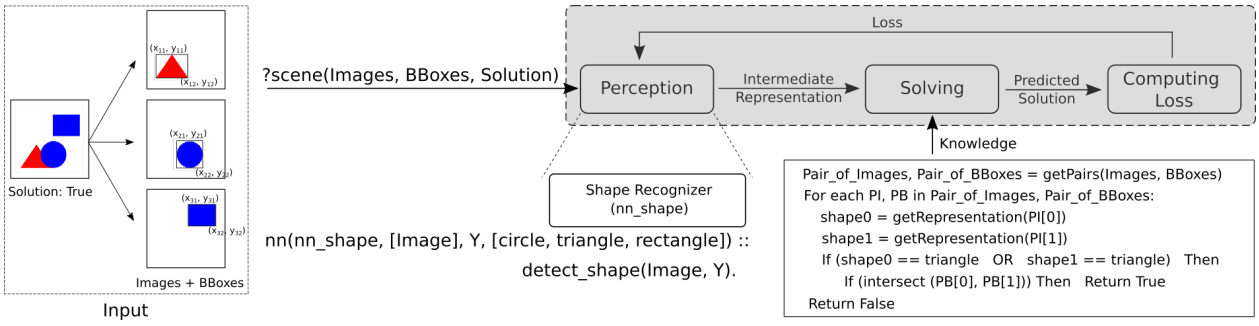

Figure 4: The DeepProbLog pipeline in visual relation recognition.

infers a suitable class label for the input image accordingly. The pseudocode of the knowledge is given in Figure 4. *getPair(Images, BBoxes)* gets the list of pairs of objects and their equivalent bounding boxes. Within an iterative process over the list of pairs, *getRepresentation(Image)* invokes the shape recognition neural network for each object in a pair (*PI[0]* and *PI[1]*). If there is at least one `triangle` predicted in a pair, the solver checks whether the pairs *intersects* or not using the information related to their bounding boxes (*PB[0]* and *PB[1]*). It returns *True* if an *intersect* with a *triangle* is identified, and returns *False* otherwise.

Experiments

Here, we compare the state-of-the-art neural models for word problem solving and visual relation recognition with our DeepProbLog models. Although it is an unfair comparison considering the additional knowledge support that DeepProbLog models possess, our intention is to show how the ability to explicitly represent and reason upon such knowledge helps in addressing generalization problems.

Word problem solving

Datasets As mentioned earlier, we collected 330 *Change* problems from the datasets, AddSub (Kushman et al. 2014) and SingleOp (Roy, Vieira, and Roth 2015). Out of the 330 problems collected, 300 were single-update problems and 30 were two-update problems. We created two test sets - the first, $Test_{similar}$ had 30 single-update problems , the second, $Test_{compositional}$ had 30 two-update problems. All the remaining (270) problems were used as the training set.

Baselines We compared the performance of DeepProbLog-based solver against the state-of-the-art neural solvers. The first baseline, **Graph2Tree** (Zhang et al. 2020), uses a graph based encoder to encode the word problem and a tree-based decoder to generate the expression tree associated with the word problem. The second baseline, is the *G*oal driven *T*ree *S*tructured neural solver - **GTS** (Xie and Sun 2019) that uses Long Short Term Memory (LSTM) to encode the word problem, and a tree-based decoder to generate the equation. The third baseline is a Sequence-to-Sequence - **Seq2Seq** network based on BiLSTM (Luong, Kayser, and Manning 2015), that translates the word problem into equation [3].

Experimental Setup All the baseline models are provided with RoBERTa pretrained embeddings ($768d$), while DeepProbLog starts with random $256d$ embeddings. The embedding layer is trained along with other layers in the network. It should be noted that, while all the baselines are trained with $(word\ problem, equation)$ pairs, DeepProbLog is trained with $(word\ problem, final\ solution)$ pairs and hence is weakly supervised. But, the knowledge provided enables it to derive equations.

Results Table 1 shows performance of the solvers on the two test sets - $Test_{similar}$ and $Test_{compositional}$. As can be observed, all the baselines fail to solve any of the problems in $Test_{compositional}$. As mentioned earlier, all the solvers are trained with problems that are mapped to equations of the form $(X = N1 +/- N2)$ unlike the problems in $Test_{compositional}$, which are mapped to equations of the form $(X = N1 +/- N2 +/- N3)$. The baseline models fail to generalize to unseen equation forms/templates. Since the perception module of DeepProbLog processes the word problem sentence-wise, learning a representation for each sentence and then composing these sentence representations based on the knowledge provided to derive the equation, DeepProbLog is able to perform better on $Test_{compositional}$, solving 70% of the problems correctly. DeepProbLog could not beat the performance of Graph2Tree or GTS on $Test_{similar}$. We believe this is because both Graph2Tree and GTS are trained on the equations directly (fully supervised), rather than on just the final solution as done in DeepProbLog.

Visual Relation Detection

Datasets To evaluate state-of-the-art methods more accurately, we built our own dataset composed of images containing geometrical shapes for a simple classification task. Each image is labeled with *True* or *False* depending on whether there is a triangle (regardless of color, shape, size) intersecting with another shape in the image or not, respectively. This allows us to form different set of data w.r.t different compositional configurations.

[3]Baseline implementations were from https://github.com/arkilpatel/SVAMP (Patel, Bhattamishra, and Goyal 2021)

Table 1: Accuracy of DeepProblog in dealing with compositionality compared to baselines (Word Problem Solving)

Test Set	Graph2Tree	GTS	Seq2Seq	DeepProbLog
$Test_{similar}$	**96.6%**	90%	73.3%	86.6%
$Test_{compositional}$	0	0	0	**70%**

Table 2: Accuracy of DeepProblog in dealing with compositionality compared to baselines (Visual Relation Recognition)

Test Set	CNN-MLP	PrediNet	ResNet-50	DeepProbLog
$Test_{similar}$	81.35%	82.0%	95.0%	**95.7%**
$Test_{compositional\_cps}$	57.5%	45.1%	72.7%	**89.3%**
$Test_{compositional\_total}$	53.43%	37.6%	58.75%	**76.4%**

The dataset contains 5000 RGB images (100×100) composed of 3 different types of shapes including *circle*, *triangle*, *rectangle*, in 3 different colors - *red*, *blue* and *green*. As an individual instance of data, each RGB image in our dataset is associated with a list of bounding boxes related to its shapes and a class label.

The training set which involves 80% of data (= 4000) is generated w.r.t the following constraints: (I) It is an even 50-50 split across the binary *True/False* class labels. (II) There are exactly two shapes in each image. (III) 90% of the positive instances (i.e., a triangle intersects with another shape) in the training data comply with the following shape/position/color constraints: *a red triangle is located at the left side of the second shape that is blue.* The remaining 10% of the positive instances do not follow the above constraint. (IV) We apply no shape/position/color constraints on negative (label = *False*) instances in the training set.

Given the above set of constraints for the training set, we have defined 3 test sets, each of size 250, as follows:

- $Test_{similar}$ is the test set with the same distribution as the training set;
- $Test_{compositional\_cps}$ is the test set where the number of shapes per image is as in the training set, however, instead of 90%, only 10% of the positive cases follow the color/position/shape (cps) constraint mentioned in (III).
- $Test_{compositional\_total}$ is the test set where the constraints (II, III) in the training set are violated (each image is composed of 3 shapes).

Baselines We compare the performance of DeepProbLog with 3 baselines. The first two are the standard neural-based networks - **CNN-MLP** and **ResNet-50** (He et al. 2016). As the third baseline we applied **PrediNet** (Shanahan et al. 2020) which, as mentioned earlier, is an end-to-end network that learns to directly map an image to propositions representing relations between objects in the image.

Experimental Setup The convolutional network used in our CNN-MLP is composed of 2 parts: (1) a 3-layer CNN network as an encoder with batch normalization and ReLU activation function trained by stochastic gradient descent and ADAM optimizer, and (2) a two-layer fully connected MLP with softmax activation function with output size 2 representing the binary classes *True* and *False* in the dataset. The perception part in our DeepProbLog implementation also uses the same configuration as mentioned above.

We configured PrediNet with a single task setting where the task refers to our relation of interest: *a triangle intersecting with another shape*. The structure of the PrediNet network evaluated in this work is equipped with a single convolutional input layer and an output multi-layer perceptron (MLP) resulting in a one-hot label denoting True or False.

The three aforementioned neural-based methods accept as input only images. In order to have a fair comparison between these methods and DeepProblog model which also relies on information about the bounding-boxes, we first preprocessed the data by augmenting each image with the bounding boxes of its shapes drawn on the image. This can assist the neural based models to recognize shapes even when same-color shapes intersect.

Results Table 2 shows performance of the models on the 3 test sets - $Test_{similar}$ and $Test_{compositional\_cps}$ and $Test_{compositional\_total}$. As can be observed, all the baselines show significant drop in performance when tested on the compositional test sets. It is not surprising to see this drop specifically in case of $Test_{compositional\_total}$ dataset as the baselines were trained on examples with images with just two objects. Our intention is to show that DeepProbLog is able to generalize better to a variable number of objects without requiring to see such examples during training. This is because DeepProbLog processes the input image object-wise, learning the shape of each object and then infers the label for the input image based on both the knowledge it has about the pattern of interest (*'Is there a triangle intersecting with another shape in the image?'*) and the geometrical knowledge regarding intersection of bounding boxes.

Discussion & Future work

We focused on compositional generalization with respect to the two specific tasks of word problem solving and visual relation recognition. Through an empirical evaluation, we showed that neuro-symbolic models are able to adapt better to distribution shifts as a result of compositionality by leveraging task specific knowledge. It is worth mentioning that a neuro-symbolic approach can offer solutions when it is difficult to collect more data but straightforward to represent knowledge useful for tackling generalization problems. Our next step is to implement models based on other NeSy frameworks and compare them with each other.

Acknowledgments

The authors would like to thank Prof. Luc De Raedt for his insightful comments and Robin Manhaeve for his help with the DeepProbLog framework. This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

Dai, B.; Zhang, Y.; and Lin, D. 2017. Detecting Visual Relationships with Deep Relational Networks. *CoRR*, abs/1704.03114.

De Raedt, L.; Dumančić, S.; Manhaeve, R.; and Marra, G. 2020. From statistical relational to neuro-symbolic artificial intelligence. *arXiv preprint arXiv:2003.08316*.

De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. ProbLog: A Probabilistic Prolog and Its Application in Link Discovery. In *IJCAI*, volume 7, 2462–2467. Hyderabad.

Ding, D.; Hill, F.; Santoro, A.; Reynolds, M.; and Botvinick, M. 2021. Attention over learned object embeddings enables complex visual reasoning. In *35th NeurIPS*.

Girdhar, R.; and Ramanan, D. 2019. CATER: A diagnostic dataset for Compositional Actions and TEmporal Reasoning. *CoRR*, abs/1910.04744.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask R-CNN. In *IEEE ICCV*, 2980–2988.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *IEEE CVPR*, 770–778.

Johnson, J.; Hariharan, B.; van der Maaten, L.; Fei-Fei, L.; Zitnick, C. L.; and Girshick, R. B. 2016. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. *CoRR*, abs/1612.06890.

Kimmig, A.; Van den Broeck, G.; and De Raedt, L. 2011. An algebraic Prolog for reasoning about possible worlds. In *25th AAAI Conf. on Artificial Intelligence*.

Klein, D.; and Manning, C. D. 2003. Accurate unlexicalized parsing. In *41st ACL*, 423–430.

Klinger, T.; Adjodah, D.; Marois, V.; Joseph, J.; Riemer, M.; Pentland, A.; and Campbell, M. 2020. A Study of Compositional Generalization in Neural Models. *arXiv preprint arXiv:2006.09437*.

Kushman, N.; Artzi, Y.; Zettlemoyer, L.; and Barzilay, R. 2014. Learning to automatically solve algebra word problems. In *52nd ACL*, 271–281.

Lake, B.; and Baroni, M. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *PMLR*, 2873–2882. PMLR.

Li, Q.; Huang, S.; Hong, Y.; Chen, Y.; Wu, Y. N.; and Zhu, S.-C. 2020. Closed loop neural-symbolic learning via integrating neural perception, grammar parsing, and symbolic reasoning. In *PMLR*, 5884–5894.

Luong, M.-T.; Kayser, M.; and Manning, C. D. 2015. Deep neural language models for machine translation. In *19th CoNLL*, 305–309.

Manhaeve, R.; Dumančić, S.; Kimmig, A.; Demeester, T.; and De Raedt, L. 2018. Deepproblog: Neural probabilistic logic programming. *arXiv preprint arXiv:1805.10872*.

Mao, J.; Gan, C.; Kohli, P.; Tenenbaum, J. B.; and Wu, J. 2019. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*.

Messina, N.; Amato, G.; Carrara, F.; Falchi, F.; and Gennaro, C. 2018. Learning Relationship-aware Visual Features. In *Proc. the European Conf. on Computer Vision Workshops*.

Patel, A.; Bhattamishra, S.; and Goyal, N. 2021. Are NLP Models really able to Solve Simple Math Word Problems? *arXiv preprint arXiv:2103.07191*.

Powell, S. R.; and Fuchs, L. S. 2018. Effective word-problem instruction: Using schemas to facilitate mathematical reasoning. *Teaching exceptional children*, 51(1): 31–42.

Roy, S.; and Roth, D. 2016. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*.

Roy, S.; Vieira, T.; and Roth, D. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3: 1–13.

Santoro, A.; Raposo, D.; Barrett, D. G.; Malinowski, M.; Pascanu, R.; Battaglia, P.; and Lillicrap, T. 2017. A simple neural network module for relational reasoning. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *NIPS*, volume 30.

Shanahan, M.; Nikiforou, K.; Creswell, A.; Kaplanis, C.; Barrett, D. G. T.; and Garnelo, M. 2020. An Explicitly Relational Neural Network Architecture. In *37th PMLR*, volume 119, 8593–8603.

Shindo, H.; Dhami, D. S.; and Kersting, K. 2021. Neuro-Symbolic Forward Reasoning. *arXiv preprint arXiv:2110.09383*.

Xie, Z.; and Sun, S. 2019. A Goal-Driven Tree-Structured Neural Model for Math Word Problems. In *IJCAI*, 5299–5305.

Yi, K.; Gan, C.; Li, Y.; Kohli, P.; Wu, J.; Torralba, A.; and Tenenbaum, J. B. 2020. CLEVRER: Collision Events for Video Representation and Reasoning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zhang, D.; Wang, L.; Zhang, L.; Dai, B. T.; and Shen, H. T. 2019. The gap of semantic parsing: A survey on automatic math word problem solvers. *IEEE transactions on pattern analysis and machine intelligence*, 42(9): 2287–2305.

Zhang, J.; Wang, L.; Lee, R. K.-W.; Bin, Y.; Wang, Y.; Shao, J.; and Lim, E.-P. 2020. Graph-to-tree learning for solving math word problems. ACL.

Zhu, M.; Zhang, Y.; Chen, W.; Zhang, M.; and Zhu, J. 2013. Fast and accurate shift-reduce constituent parsing. In *51st ACL*, 434–443.