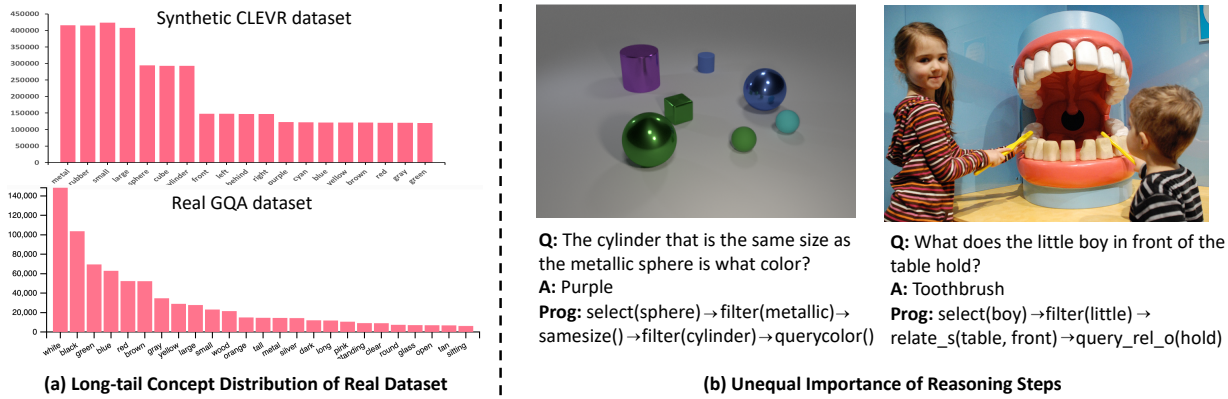
Calibrating Concepts and Operations: Towards Symbolic Reasoning on Real Images

Zhuowan Li[1] Elias Stengel-Eskin[1] Yixiao Zhang[1] Cihang Xie[2]
Quan Tran[3] Benjamin Van Durme[1] Alan Yuille[1]
[1]Johns Hopkins University [2] University of California, Santa Cruz [3]Adobe Research



(a) Long-tail Concept Distribution of Real Dataset (b) Unequal Importance of Reasoning Steps

Figure 1: Statistics and examples from the synthetic CLEVR dataset and the real GQA dataset. Compared to the synthetic dataset, VQA on real data needs to deal with long-tail concept distribution and uneven importance of reasoning steps.

Abstract

While neural symbolic methods demonstrate impressive performance in visual question answering on synthetic images, their performance suffers on real images. We identify that the long-tail distribution of visual concepts and unequal importance of reasoning steps in real data are the two key obstacles that limit the models' real-world potentials. To address these challenges, we propose a new paradigm, ***Calibrating Concepts and Operations (CCO)****, which enables neural symbolic models to capture underlying data characteristics and to reason with hierarchical importance. Specifically, we introduce an executor with learnable concept embedding magnitudes for handling distribution imbalance, and an operation calibrator for highlighting important operations and suppressing redundant ones.*

Our experiments show CCO substantially boosts the performance of neural symbolic methods on real images. By evaluating models on the real world dataset GQA, CCO helps the neural symbolic method NSCL outperforms its vanilla counterpart by 9.1% (from 47.0% to 56.1%); this result also largely reduces the performance gap between symbolic and non-symbolic methods. Additionally, we create a perturbed test set for better understanding and analyzing model performance on real images. Code is available at https://lizw14.github.io/project/ccosr.

1. Introduction

Visual question answering (VQA) aims to develop a model that can answer open-ended questions from images. Currently, end-to-end methods, which directly make predictions over dense visual and textual features [37, 19], represent the most effective class of models for VQA. Nonetheless, such methods have been criticized for exploiting shortcuts (*e.g.*, statistical dataset bias [1, 11], question prior [2] or isolated text and image elements [25]) to answer questions; these shortcuts often make them unable to generalize well on out-of-domain data.

In contrast, neural symbolic methods [5, 18, 38, 26] are equipped with strong reasoning ability, enabling them to answer multi-hop and complex questions in a compositional and transparent manner—they first parse each question into a program with a series of reasoning steps, and then compose neural modules on the fly to execute the program on the image. While symbolic methods achieve nearly perfect performance on synthetic dataset, they perform poorly on real-world datasets. For instance, neural symbolic concept learner (NSCL) [26] achieves 98.9% accuracy on the synthetic CLEVR dataset [17], but only 47.0% accuracy on the real-world GQA dataset [15]. Note the original NSCL cannot be directly applied to GQA; this 47.0% accuracy is obtained from our own re-implementation, where minimal but

necessary modifications are made (*e.g.*, adding in $same$ and $common$ modules), for making models runnable on GQA.

As summarized in Figure 1, we note there are two major differences between synthetic datasets and real-world datasets. First, while visual concepts are well-balanced in the synthetic datasets, they follow a long-tail distribution in real-world datasets. For example, as shown in Figure 1(a), in GQA, common concepts like "man", "window", "black", "white" are far more frequent than uncommon ones like "pink" and "eraser", in both questions and answers. Second, unlike in synthetic data, the reasoning steps on real data have varying importance, mainly because of redundancy/over-specification in question description. For example, as shown in Figure 1(b), in the question "What is the little boy doing?", the noun (*i.e.*, boy) itself is enough to select the person being asked about while the adjective (*i.e.*, little) only serves as a nuisance factor.

We identify that this mismatch of dataset characteristics is the main obstacle for adapting neural symbolic methods from synthetic datasets to real-world datasets. More concretely, we find that the original architecture designs of neural symbolic methods (which were designed/verified mainly on synthetic datasets) are no longer suitable for the real-world setting. For examples, as shown in Section 3, even simple operations like removing the normalization on concept embeddings or manually assigning larger weights to less discriminative modules are effective to improve the performance of neural symbolic methods on real images.

To better cope with real images, we propose *Calibrating Concepts and Operations (CCO)*, which enables neural symbolic methods to explicitly learn weights for concept embedding and reason with contextual module importance. Specifically, CCO learns different concept embedding magnitudes for each execution module, and learns an operation weight predictor to contextually predict weights for each operation in the reasoning program. In this way, the model will be able to handle unbalanced concept distributions and to reason with varying operation importance.

Our empirical results show that CCO substantially boosts the applicability of neural symbolic methods on real images. For example, on the real-world GQA dataset, CCO outperforms the baseline NSCL by a large margin of 9.1% (from 47.0% to 56.1%). Moreover, the proposed CCO largely reduces the performance gap between the symbolic method and the state-of-the-art non-symbolic methods [32, 16] on real-world GQA dataset.

Additionally, based on the proposed operation weight calibrator, we create a perturbed test set by progressively removing the operations with low weights from testing questions. Our purpose is to verify whether the learned operation weights are able to highlight important operations and suppress redundant ones, and simultaneously to access the robustness of different models regarding this operation information erasing. Our analysis reveals 1) GQA questions contain superfluous information by way of over-specification and 2) the ability to effectively handle this extraneous information is crucial for models to improve performance. We hope this perturbed test set can facilitate researchers to better understand the compositionality of VQA questions and to further improve symbolic reasoning over real images.

2. Related Work

Visual Question Answering (VQA) [6] requires an understanding of both visual and textual information. Pure deep learning methods that based on convolution, LSTM and attention have achieved good performance. For example, Fukui *et al*. [10] used multimodal compact bilinear pooling to combine visual and language features into a joint representation. Yang *et al*. [37] used stacked attention to refine the attended image region relevant to the question. Kim *et al*. [19] proposed bilinear attention network to learn attention between the two modalities with residual connections between multiple attention maps. Yang *et al*. [36] proposed a tiered relational reasoning method that dynamically attends to visual objects based on textual instruction.

Visual reasoning. Prior work has suggested that above mentioned VQA models may rely on dataset shortcuts and priors to predict answer [1, 11, 2, 29, 7, 8]. Therefore, recent efforts have focused more on visual reasoning with complex compositional questions that requires multi-step reasoning and true visual scene understanding. Johnson *et al*. [17] propose CLEVR that requires reasoning over synthetic scenes with compositional questions automatically generated using question templates. Hudson *et al*. [15] further constructed GQA, a dataset with real images and procedurally generated multi-step questions, for visual reasoning.

Attention is widely used in vision and language tasks, including image captioning [34, 23, 24, 22], visual question answering [37, 19, 36], referring expressions [39, 35, 33]. It is shown effective in learning distinct importance of images in an image group, of sub-regions over an image or of words in a sentence. Our work calibrates different concepts and operations, thus enables the model to reason with weighted concepts and contextual operation importance.

Neural symbolic methods. [42, 43, 41] show impressive reasoning ability on abstract reasoning tasks like [40, 46]. For VQA, Andreas *et al*. [5] proposes neural modular networks, which decomposes a question into a functional program (reasoning steps) that can be executed by neural modules over the image. This method gets further improve by executing the functional program explicitly [18, 27, 14, 16, 9] or implicitly [13, 12], manipulating visual and textual features using convolution or dual attention. Specifically, [38, 26, 21] propose a pure symbolic executor given pre-parsed or learned explicit programs, and

achieve state-of-the-art performance on CLEVR. Recently, Amizadeh *et al*. [3] propose a symbolic reasoner based on first order logic to diagnose reasoning behavior of different models. While symbolic methods provide interpretable programs, their reasoning capacity on real data is still limited [15]. Our work aims to reduce the performance gap between symbolic and non-symbolic models on real data.

3. Motivation

In this section, we provide simple examples to demonstrate how the dataset differences (between the synthetic CLEVR and the real GQA) affect the performance of neural symbolic methods. Interestingly, we find that the traditional design principles in neural symbolic methods, which are usually obtained from synthetic datasets, may not be optimal for the real-world datasets.

3.1. Normalized Concept Embedding?

For neural symbolic methods, at each step of execution, a similarity score between each object embedding and the learned concept semantic embedding is computed to select the target object that is being asked about (*i.e*., selecting the object that is closest to the query concept) and to predict answers (*i.e*., selecting the concept that is closest to the target object). By default, normalization is applied to both object embedding and concept embedding.

Interestingly, on the real-world GQA, we find this default strategy is not optimal; simply removing the normalization on concept embedding yields substantially better performance (+3.4%). This phenomenon indicates that in addition to the angle alignment between object embedding and concept embedding, the magnitude of concept embedding is also informative for symbolic reasoning on real images.

We conjecture this is because the magnitude can represent the concept distribution, which is drastically different between synthetic datasets and real datasets. For example, while CLEVR contains only a relatively small and perfectly-balanced set of concepts (*i.e*., 19 concepts including shapes, materials), real datasets deal with thousands of concepts which are far more complex and follows a long-tail distribution. We validate this hypothesis in Section 6—with a learnable magnitude for each concept embedding, we find its value is strongly correlated with concept frequency, *i.e*., more frequent concepts tend to have larger magnitudes.

3.2. Module Re-weighting

In addition to this long-tailed distribution, the reasoning steps on real data are of varying importance during execution. For example, in most cases, the $select(noun)$ module are more discriminative than the $filter(attribute)$ or the $relate(relationship)$ operations, due to implicit entailment in natural language and over-specification of the question (*e.g*., "little boy", "trees below the sky"). Therefore directly adapting symbolic methods to GQA will bias the model towards putting more focus on learning discriminative operations while neglecting the rest, resulting in errors on questions where all operations are important. For example, in Figure 2, the question asks for a bag that is not black; but $select(bag)$ operation produces large values, overriding the $filter(not\ black)$ step, leading to a "yes" answer, even though the bag is NOT in the required color.

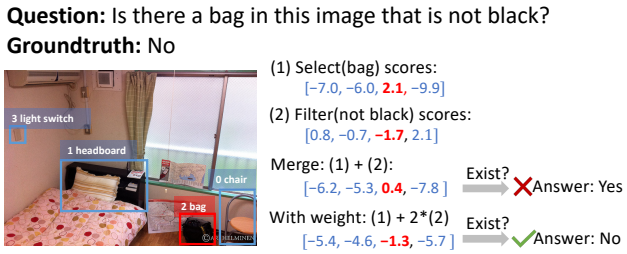


Figure 2: A failure case that can be corrected by re-weighting the operations. The $select(bag)$ operation overrides $filter(not\ black)$, thus lead to incorrect answer. This can be corrected by scaling up the result of $filter$ operation.

Surprisingly, in this example, if we simply magnify the output of $filter(not\ black)$ operation by a factor of 2, the $filter$ operation then can successfully rule out the black bag, thus correctly answering the question. This result suggests that, while many questions contain redundant operations that the model tends to overlook, correctly re-weighting the operations is crucial for symbolic reasoning on real images.

4. Calibrating Concepts and Operations

Given the observations in Section 3, we next explore designing more sophisticated algorithms for automatically and effectively dealing with the complex characteristics of real data (*e.g*., long-tailed distribution and unequal reasoning steps), for the purpose of increasing neural symbolic methods' real-world applicability.

4.1. Formulation

In symbolic reasoning, a parser first parses a question $Q =< \hat{w}_1, ..., \hat{w}_l >$ into a tree-structured functional program P. The program P consists a set of modules $< p_1, ..., p_m >$ with dependency relationships between each other. As the functional program is either a chain or a binary tree, it can be linearized into sequence by pre-order traversal. Each operation p has its type p^t (e.g., $select$, $filter$), attribute p^a (e.g., $color$, $material$) and concept p^c (e.g., red, $plastic$). We denote the total number of module types, attributes and concepts as n_t, n_a, n_c, respectively. Then execution modules are composed on the fly, based on this generated program P. The module outputs are merged based on their dependency relationship and fed into the final module to get the answer $\mathbf{a}$.

For scene representation, we first obtain a set of feature vectors $\mathbf{v}_i \in \mathbb{R}^d$ from the image I, with n objects detected in the image. Specifically, the feature vector $\mathbf{v}$ can be either visual features obtained from Faster RCNN [30], or the symbolic representation for each object (which can be obtained by concatenating distributions over N_c object categories and N_a attributes).

4.2. Basic Executor Architecture

Given the program P, the executor then executes it over input scene representations $\mathbf{v}$ to get a reasoning answer $\mathbf{a}$. The basic executor principle follows the design in [26].

As shown in Figure 3, each module (except for the output module) produces a distribution $\mathbf{d}$ over N objects in the image ($\mathbf{d} \in \mathbb{R}^N$), which are then merged based on their dependencies. In contrast to the default setup in the synthetic dataset, we use the mean operation (rather than minimum as in NSCL) to merge the module results due to its more stable training behavior. Finally, the output module takes in the object distribution produced by intermediate modules and queries/verifies the specified attribute of the selected object.

For module design, a semantic embedding $\mathbf{c}$ is learned for each concept (*e.g.*, man, red, round, etc). Without loss of generality, we illustrate the computation for $select$ and $query$ modules. The architecture of the other module types can be found in supplementary materials.

We take the module $select[name](spectator)$ as an example. First, a small network $\mathcal{M}_{name}$ maps each object representation $\mathbf{v}_i$ into the concept embedding space, and then the similarity $\mathbf{s}_i$ between the embedded object representation $\mathbf{e}_i$ and the embedding of concept "spectator" ($\mathbf{c}_{spectator}$) is computed. This similarity $\mathbf{s}_i$ can be interpreted as the likelihood of each object being "spectator". The computation of $select$ module can be summarized as the following:

$$\mathbf{e}_i = \mathcal{M}_{attr}(\mathbf{v}_i) \tag{1}$$
$$s_i = \text{sim}(\mathbf{e}_i, \mathbf{c}_{cept}) \tag{2}$$
$$\mathbf{d}_{select} = [s_1, s_2, ..., s_N] \tag{3}$$

where cosine similarity, *i.e.*, dot product of normalized $\mathbf{e}$ and $\mathbf{c}$, is used for similarity computation.

The detailed network architecture of the representation mapping network $\mathcal{M}_{attr}$ is shown in Figure 3. It gates the input object representation and passes it through a MLP to get the corresponding semantic embedding. The semantic embedding is then added with spatial embedding to get the final object embedding. The mapping networks $\mathcal{M}$ corresponding to different attributes share the same network architecture but with different weights.

We also briefly summarize the computation of $query$ module below, as another example:

$$\mathbf{e}_i = \mathcal{M}_{attr}(\mathbf{v}_i) \tag{4}$$
$$\mathbf{e} = \mathbf{d} \cdot [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_N] \tag{5}$$
$$\mathbf{a}_j = \text{sim}(\mathbf{e}, \mathbf{c}_j) \tag{6}$$

where the operation $\cdot$ refers to element-wise multiplication between two vectors, and $\mathbf{c}_j$ refers to the concept embedding of possible answers.

4.3. Calibrating Concepts and Operations

We hereby formally propose *Calibrating Concepts and Operations (CCO)*, which includes a concept calibration module and an operation calibration module, to help neural symbolic methods improving their applicability on real images. The overall design is illustrated in Figure 3.

Calibrating concepts. As diagnosed in Section 3, the magnitude of concept embedding $\mathbf{c}$ is informative for measuring the similarity between the object embedding and concept embedding. This motivates us to design an extra architectural element for explicitly capturing such information in magnitudes. Moreover, this designed architectural element is expected to be adaptive for different concepts, as each distinct type of operation is dealing with varying concept frequency distributions. For example, general concepts like "person" are common in $select$ module, but not in $query$ as the answers usually expect more specific concepts.

In light of these intuitions, we offer a simple solution—explicitly learning different embedding magnitudes for each module type. We expect the learned norm sizes can encode the concept distribution, thus the more frequent concept has larger norms size and lead to larger similarity values. Concretely, we calibrate concept embeddings by:

$$\mathbf{c}_{concept} = w^{type}_{concept} \mathbf{c}_{concept} \tag{7}$$

where w is different for each module type and each concept. This is applied whenever concept embeddings are used for similarity computation (*e.g.* in Equation 2). To this end, distinct types of modules share the same concept embedding direction, but varying magnitudes, corresponding to different concept distributions.

Calibrating operations. As shown in Section 3, on real images, it is important to enable the model to reason with different operation importance. To this end, we propose to customize the weight of each operation in the program. Specifically, a bi-directional LSTM weight predictor is used here to predict operation weights based on the whole program. For each operation p_i in the program, its weight w_i is computed as following:

$$\mathbf{e}_i = [\mathbf{e}^t_i; \mathbf{e}^a_i; \mathbf{e}^c_i] \tag{8}$$
$$\mathbf{h}_1, ..., \mathbf{h}_m = \text{LSTM}(\mathbf{e}_1, ..., \mathbf{e}_m) \tag{9}$$
$$w_i = \text{sigmoid}(W\mathbf{h}_i) \tag{10}$$

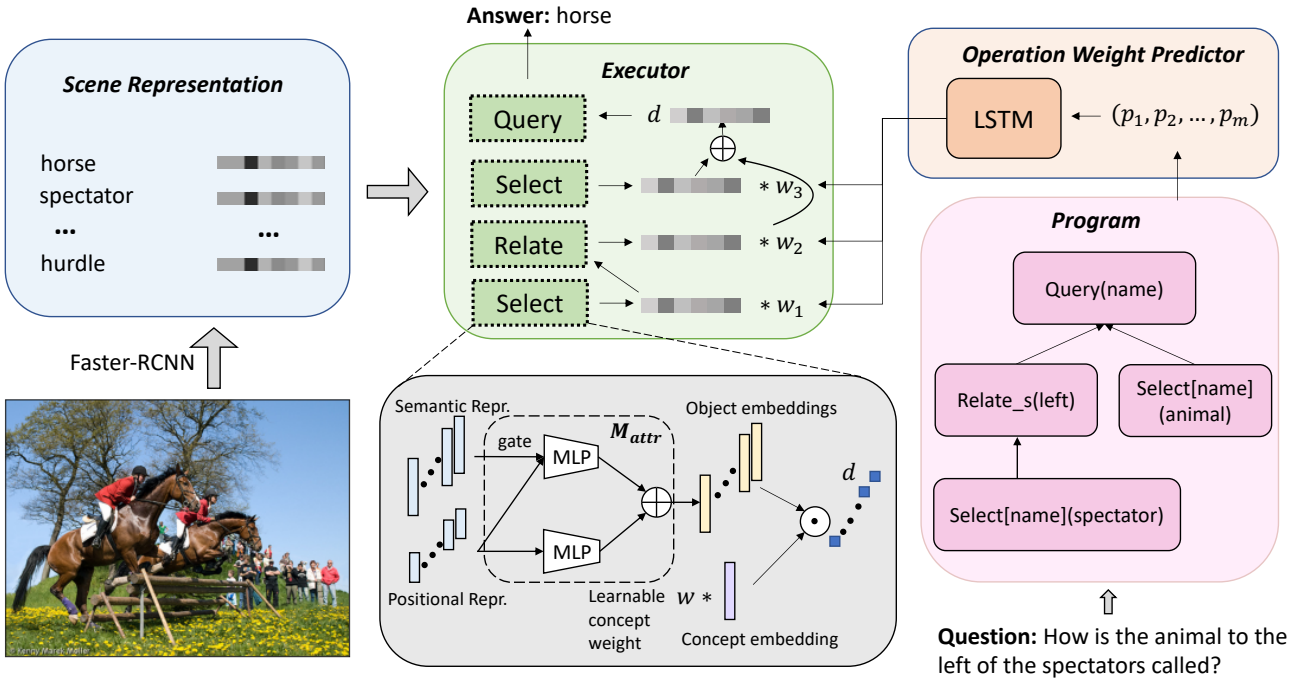


Figure 3: Overview of our method. We first parse the image into a symbolic scene representation in the form of objects and attributes, then parse the question into a program. In each reasoning step, a reasoning module takes in the scene representation and the instruction from the program, and outputs a distribution over objects. The Operation Weight Predictor predicts a weight for each reasoning module, which will be used to merge module outputs based on the program dependency. The final distribution is fed into the output module to predict answers.

where m is the program length. The inputs $\mathbf{e}$ to LSTM is the concatenation of the operation type embedding $\mathbf{e}^t$, the attribute embedding $\mathbf{e}^a$ and the concept embedding $\mathbf{e}^c$. The predicted operation weights are then used to merge outputs of the operations with a weighted-sum operation:

$$\mathbf{d}_i = \sum_{j \in \mathcal{D}(p_i)} w_j \mathbf{d}_j \tag{11}$$

where $\mathcal{D}(p_i)$ is the set of dependency operations of operation p_i. In this way, operations with higher weights play a more important role in the merging step.

Summary. With the proposed CCO, neural symbolic executor now is able to capture the underlying data characteristics and reason with learnable operation importance. As we will next, CCO substantially boosts model performance on GQA, meanwhile largely reduces the performance gap between symbolic and non-symbolic methods.

5. Experiments

5.1. Dataset and Experiment Setup

Dataset. Our experiments are on GQA [15], which is a dataset focusing on reasoning and compositional question answering over real images. Building on top of Visual Genome dataset [20], it contains more than 110K images and 22M questions. Each image is annotated with a scene graph cleaned from Visual Genome that contains the information of objects, attributes and relationships. Each question comes with a corresponding functional program that specifies reasoning steps. By default, we use its balanced version with 943k, 132k, 13k and 95k questions in train, val, testdev and test split for training and evaluating.

Scene representation. We train a Faster RCNN with an additional attribute head using cross entropy loss following [4]. We train with 1313 object classes (lemmatize and remove plurals) and 622 attributes. The model gets 24.9 mAP for object detection and 17.1 groundtruth attribute average rank.[1] The 1935-d concatenation of class and attribute scores are used as symbolic scene representation.

Implementation details. The inner dimension of our model is 300d. The concept embedding is initialized using GloVe embedding [28]. We train our reasoning model using the Adam optimizer with an initial learning rate of 0.0005 and a batch size of 256. Linear learning rate is used with 2000 warm-up steps. We train the model for a total of

[1] Attribute prediction is evaluated by the average rank of groundtruth attribute in all the 622 attributes. We only consider the correctly detected objects (IOU>0.5) for attribute evaluation.

30 epochs, with early stopping (based on accuracy on the balanced testdev split) to prevent overfitting. To avoid confounding caused by parsing errors, we use *gold programs* to analyze the execution performance by default.

5.2. Execution Results

	Concept	Operation	Acc.
1 (Baseline)	Normalized	Average	47.01
2	Normalized	Calibrated	51.30
3	Unnormalized	Calibrated	54.65
4 (Ours)	**Calibrated**	**Calibrated**	**56.13**

Table 1: Accuracy comparison on the balanced GQA testdev split. Compared to the baseline, both concept calibration and operation calibration substantially improve model performance. The best performance is achieved by calibrating both concept and operation.

We choose NSCL [26] as our baseline model. By default, concept embeddings are normalized before similarity computation (cosine similarity) and operation results are merged by taking the average. After appplying minimal but necessary changes to NSCL for making it runnable on GQA, it achieves 47.01% accuracy. We then integrate the proposed concept and operation calibration strategies on top of this baseline, while keeping other settings unchanged. As shown in the fourth row of Table 1, CCO helps the baseline gain a substantial improvement, *i.e.*, the accuracy is increased from 47.01% to 56.13%. This 9.12% improvement margin in accuracy demonstrates the effectiveness of our proposed method.

To further analyze the improvement brought by each individual component, we progressively add in our proposed concept calibration and operation calibration into the NSCL baseline. As shown in the second row of Table 1 where the operation calibration is added, it outperforms the baseline by 4.29%, demonstrating the effectiveness of operation calibration. We then remove the normalization of concept embeddings and keep the embedding magnitudes when computing similarity. As shown in the third row of Table 1, such strategy successfully leads to an additional 3.35% improvement. This result suggests that the embedding magnitudes are informative, which is consistent with our analysis in Section 3.1. In summary, these results support that both concept weighting and operation weighting are useful for improving the NSCL baseline.

5.3. Ablations

Scene representations. Regarding scene representations, besides using symbolic representations, we also test model performance with other alternatives. To validate the correctness of our model design, we feed the operation modules with gold scene representation. Our CCO achieves 89.61% accuracy, which is similar to human performance (89.30%). This high upper bound indicates that model performance can be further improved by better visual perception.

We also examine the model performance by using visual features (Faster-RCNN feature after mean-pooling) as scene representation. Our CCO achieves 53.00% accuracy, where the 3.13% performance gap (*i.e.*, 53.00% *vs*. 56.13%) shows the advantage of the abstract symbolic scene representation over the dense visual features.

Program parsing. In all previous experiments, we apply gold program for facilitating performance analysis. While in this part, we now examine the model performance in the wild, *i.e.*, gold program is no longer available. In order to parse the question into functional program, we apply MISO, a popular sequence-to-graph parser used for parsing in a number of graph-based formalisms [44, 45, 31]. Different from simple sequence-to-sequence parser as in [18] that can only handle program with one argument, or the two-stage parser as in [9] that handles multiple arguments by hard constraints, MISO can automatically handle multiple arguments by treating the program as a graph. The input to MISO parser are word embedding sequences and output is a pre-order traversal of a program trees.

We present the parsing results in Table 2. We use exact match score, which is calculated by the percentage of predicted programs that exactly match the gold program, for measuring the quality of the predicted program. Compared to the parser in MMN [9], ours outperforms it by a large margin of 6.05% in terms of exact match score. Nonetheless, interestingly, we find final model accuracy is less impacted by the quality of program—by executing either ours or MMN's predicted program, the difference in the final model accuracy is only 0.1%. This seemly "frustrating" result may suggest the performance of other components in current neural symbolic methods are severely lagged behind therefore are not able to cope with the advances brought by our strong parser.

Model	Exact match	Acc.
MMN [9]	85.13	54.01
Ours	**91.18**	**54.11**

Table 2: Parsing performance on testdev_balanced, measured by exact match score and execution accuracy.

Comparing to the state-of-the-arts. To fairly compare different methods on GQA, we follow the training setups in [3, 9] where we first train the model on unbalanced training split then finetuned on balanced training split. Gold programs are used for training while parser predicted programs are used for evaluation. Performance is reported using the official evaluation metrics, including overall accuracy, accuracy on binary questions, accuracy on open questions, consistency, plausibility, validity and distribution.

	Method	Acc	Binary	Open	Consistency	Plausibility	Validity	Distribution
Non-Symbolic	LXMERT [32]	60.33	77.16	45.47	89.59	84.53	96.35	5.49
	NSM [16]	63.17	78.94	49.25	93.25	84.28	96.41	3.71
	MMN [9]	60.83	78.90	44.89	92.49	84.55	96.19	5.54
Symbolic	∇-FOL [3]	54.76	71.99	41.22	84.48	-	-	-
	CCO (ours)	56.38	74.83	40.09	91.71	83.76	95.43	6.32

Table 3: Comparison with state-of-the-art symbolic and non-symbolic methods on the official testing split.

We hereby consider three non-symbolic methods (*i.e.*, LXMERT [32], NSM [16], MMN [9]) and one symbolic method (*i.e.*, ∇-FOL [3]) for performance comparison. In short, LXMERT is a representative multi-modal pretraining method; NSM is a graph-based model that achieves state-of-the-art performance on GQA; MMN is a modular method but is still based on dense features manipulation; ∇-FOL[2] is a symbolic method based on first order logic and contextual calibration. We summarize the model performance on the held-out test split in Table 3.

Compared to the previous state-of-the-art symbolic method ∇-FOL, our proposed CCO surpasses it by 1.58% in terms of accuracy. Moreover, as shown in Table 3, we note the performance gain over ∇-FOL is mainly on the binary questions (+2.84) and on predicting consistent answers for different questions (+7.2%).

We next compare with the state-of-the-art non-symbolic methods. Though our model still has lower accuracy than these non-symbolic methods, we note their performance on consistency, plausibility and validity is on a par with each other. We conjecture this is due to the symbolic nature of our model, *i.e.*, the proposed CCO execute strictly according to the program, thus answers are plausible and valid, and questions with same underlying program get consistent answer. These results suggest that the proposed CCO largely reduces the performance gap between symbolic and non-symbolic methods on the real-world GQA dataset.

6. Analysis

6.1. Learned Embedding Magnitudes

To verify our motivation that the learned concept embedding magnitudes are informative for representing the unbalanced concept distribution in real dataset, we visualize the correlation between concept counts and their magnitude after calibration (in $query$ module), *i.e.*, $\|\mathbf{c}_{concept}\|_2$ after calibration in Equation 7. In the plot, X-axis is the count of concepts in $query$ module (taking log), and Y-axis is the learned magnitude of concept embeddings.

As verified in Figure 4, more frequent concepts consistently learn larger magnitudes, while less frequent concepts generally have smaller magnitudes. With larger magnitudes, the frequent concepts will produce values with higher confidence when computing similarity in the output of each module. Another interesting observation is that the magnitudes for few-shot concepts are not very consistent (*i.e.*, have larger variance), which is possibly caused by the insufficient number of training examples.

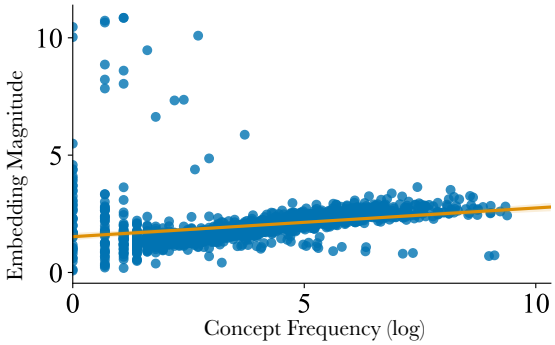


Figure 4: A positive correlation between learned embedding magnitude and concept frequency confirms our motivating intuition: more frequent concepts have larger magnitudes.

6.2. Perturbed Test Set

We create a perturbed testing data splits for the following purposes: a) we want to validate that proposed operation weighting strategy predicts larger weights for more important operations and smaller weights for unimportant ones; b) we need a test set for better studying the question overspecification in GQA dataset; and c) we aim to benchmark behavior of symbolic and non-symbolic methods in terms of how much information in the over-specified operations can be effectively utilized.

Specifically, this perturbed test set is created using the operation weights predicted by the learned LSTM operation weight predictor. We perturb the functional programs in balanced testdev splits by progressively removing the removable operations with smaller predicted weights[3]. Note that removable operations here refer to the intermediate operations that can be removed without syntactically breaking the programs, *i.e.*, $filter$, $relate$ and their dependent operations. Then, we train a simple sequence-to-sequence gen-

[2] ∇-FOL does not report full result on the official test split, therefore results on balanced testdev split is shown for comparison.

[3] We set the weight thresholds to be $-\infty$, -2, -1, -0.5, 0, $+\infty$; resulting in removing 0%, 14%,31%,70%, 90%, 100% of removable operations, respectively

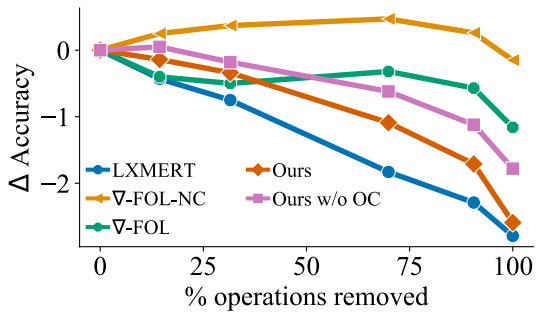


Figure 5: Accuracy drop of different models when the testing questions are progressively perturbed by removing reasoning operations with low weights.

erator to recover questions from the perturbed programs.

The results are shown in Figure 5. We test five methods, including non-symbolic method LXMERT [32], symbolic method ∇-FOL [3], its variant ∇-FOL-NC which is a pure reasoner based on first order logic, our model, and ours without operation calibrating.[4] Our observations can be summarized as the following:

Validity of operation weights. All curves exhibit a sharper decrease at the end when more operations with higher weights are removed. In other words, the removal of operations with larger predicted weights will result in bigger negative influence on model accuracy. This validates the predicted weights correctly represent operation importance.

Question over-specification. From the curves, we note while 59.0% questions in the balanced testdev split contain removable operations and are perturbed, less then 3.0% questions are incorrectly answered after removing those modules. This phenomenon suggests that for most questions in GQA dataset, the $filter$ and $relate$ operations are not necessary for figuring out the answer, *i.e.*, removing all the intermediate attributes and relationships from questions does not change the answer for most of the questions.

Effectiveness of operation weighting. Interestingly, the performance of the pure logic reasoner ∇-FOL-NC and our model without operation weighting even gets slightly increased when removing a small amount of operations. This phenomenon indicates that those operations are hard for models to learn thus can even derail the model predictions. This verifies our motivation for designing operation calibration as it helps the learning of $filter$ and $relate$ modules.

Comparison of symbolic and non-symbolic methods. Compared to symbolic methods, non-symbolic methods have larger accuracy drops, therefore indicating they can more effectively utilize the information in adjectives and relationships. Moreover, methods with higher performance tend to have larger decrease when questions are perturbed. This suggests enhancing the model's ability to understand filtering adjectives and relationships is crucial for improving symbolic methods on real images.

[4]Original accuracy of the five models (LXMERT, ∇-FOL, ∇-FOL-NC, ours, and ours w/o OC) are 58.13, 54.02, 51.86, 56.13, 55.49, respectively.

threshold	All	Easy	Hard
$-\infty$(orig)	56.13	78.03	37.42
-2	-0.14	0.4	-2
-1	-0.34	0.13	-2.17
0.5	-1.09	-0.51	-3.04
0	-1.71	-0.93	-3.86
$+\infty$	-2.59	-1.88	-4.72

Table 4: Model accuracy on perturbed easy/hard splits.

6.3. Hard and Easy Subset

We additionally perturb the visual-hard and the visual-easy testing splits [3] and evaluate our CCO model on them. Specifically, the easy split contains questions that visually easy thus can be answered correctly by their differentiable first-order logic formula, while the hard split are harder in perception. In other words, the easy split contains questions that can be answered by a perception system alone, while the hard split contains images requiring more reasoning. With perturbed versions, we can investigate to what degree low-weight operations are implicated in multi-step reasoning for visually hard questions.

We summarize the model performance in Table 4. With more operations get removed, the accuracy drop on perturbed hard split is much larger than the easy split. This indicates that the visually hard questions force the model to better utilize every piece information in the question, while easy questions contain more redundant operations that are not necessarily needed.

7. Conclusion

To improve symbolic reasoning for VQA on real images, we propose to calibrate concepts and operations (CCO), which helps models handle the unbalanced concept distribution and unequal importance of reasoning operations. Experimental results demonstrate the effectiveness of the proposed method, where CCO outperforms several baselines by a large margin and reduces the performance gap between symbolic and non-symbolic methods. Additionally, we propose a perturbed test set for better understanding and analyzing model performance on real images. We hope this dataset can help researchers to further study the potential of symbolic reasoning on real images in the future.

Acknowledgements

This work was supported by NSF #1763705 and IARPA BETTER (2019-19051600005). Elias Stengel-Eskin is supported by an NSF Graduate Research Fellowship. Cihang Xie is supported by a gift grant from Open Philanthropy.

References

[1] Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. Analyzing the behavior of visual question answering models. *arXiv preprint arXiv:1606.07356*, 2016. 1, 2

[2] Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. Don't just assume; look and answer: Overcoming priors for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4971–4980, 2018. 1, 2

[3] Saeed Amizadeh, Hamid Palangi, Alex Polozov, Yichen Huang, and Kazuhito Koishida. Neuro-symbolic visual reasoning: Disentangling. In *International Conference on Machine Learning*, pages 279–290. PMLR, 2020. 3, 6, 7, 8

[4] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086, 2018. 5

[5] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48, 2016. 1, 2

[6] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015. 2

[7] Remi Cadene, Corentin Dancette, Hedi Ben-Younes, Matthieu Cord, and Devi Parikh. Rubi: Reducing unimodal biases in visual question answering. *arXiv preprint arXiv:1906.10169*, 2019. 2

[8] Long Chen, Xin Yan, Jun Xiao, Hanwang Zhang, Shiliang Pu, and Yueting Zhuang. Counterfactual samples synthesizing for robust visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10800–10809, 2020. 2

[9] Wenhu Chen, Zhe Gan, Linjie Li, Yu Cheng, William Wang, and Jingjing Liu. Meta module network for compositional visual reasoning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 655–664, 2021. 2, 6, 7

[10] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*, 2016. 2

[11] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6904–6913, 2017. 1, 2

[12] Ronghang Hu, Jacob Andreas, Trevor Darrell, and Kate Saenko. Explainable neural computation via stack neural module networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 53–69, 2018. 2

[13] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 804–813, 2017. 2

[14] Drew A Hudson and Christopher D Manning. Compositional attention networks for machine reasoning. *arXiv preprint arXiv:1803.03067*, 2018. 2

[15] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6700–6709, 2019. 1, 2, 3, 5

[16] Drew A Hudson and Christopher D Manning. Learning by abstraction: The neural state machine. *arXiv preprint arXiv:1907.03950*, 2019. 2, 7

[17] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017. 1, 2

[18] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2989–2998, 2017. 1, 2, 6

[19] Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. Bilinear attention networks. *arXiv preprint arXiv:1805.07932*, 2018. 1, 2

[20] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017. 5

[21] Qing Li, Siyuan Huang, Yining Hong, and Song-Chun Zhu. A competence-aware curriculum for visual concepts learning via question answering. In *European Conference on Computer Vision*, pages 141–157. Springer, 2020. 2

[22] Zhuowan Li, Quan Tran, Long Mai, Zhe Lin, and Alan L Yuille. Context-aware group captioning via self-attention and contrastive features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3440–3450, 2020. 2

[23] Chenxi Liu, Junhua Mao, Fei Sha, and Alan Yuille. Attention correctness in neural image captioning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017. 2

[24] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 375–383, 2017. 2

[25] Varun Manjunatha, Nirat Saini, and Larry S Davis. Explicit bias discovery in visual question answering models. In *Pro-*

ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9562–9571, 2019. 1

[26] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*, 2019. 1, 2, 4, 6

[27] David Mascharka, Philip Tran, Ryan Soklaski, and Arjun Majumdar. Transparency by design: Closing the gap between performance and interpretability in visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4942–4950, 2018. 2

[28] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 5

[29] Sainandan Ramakrishnan, Aishwarya Agrawal, and Stefan Lee. Overcoming language priors in visual question answering with adversarial regularization. *arXiv preprint arXiv:1810.03649*, 2018. 2

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 4

[31] Elias Stengel-Eskin, Aaron Steven White, Sheng Zhang, and Benjamin Van Durme. Universal decompositional semantic parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8427–8439, 2020. 6

[32] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019. 2, 7, 8

[33] Peng Wang, Qi Wu, Jiewei Cao, Chunhua Shen, Lianli Gao, and Anton van den Hengel. Neighbourhood watch: Referring expression comprehension via language-guided graph attention networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1960–1968, 2019. 2

[34] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015. 2

[35] Sibei Yang, Guanbin Li, and Yizhou Yu. Dynamic graph attention for referring expression comprehension. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4644–4653, 2019. 2

[36] Xiaofeng Yang, Guosheng Lin, Fengmao Lv, and Fayao Liu. Trrnet: Tiered relation reasoning for compositional visual question answering. 2

[37] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 21–29, 2016. 1, 2

[38] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *arXiv preprint arXiv:1810.02338*, 2018. 1, 2

[39] Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. Mattnet: Modular attention network for referring expression comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1307–1315, 2018. 2

[40] Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. Raven: A dataset for relational and analogical visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[41] Chi Zhang, Baoxiong Jia, Mark Edmonds, Song-Chun Zhu, and Yixin Zhu. Acre: Abstract causal reasoning beyond covariation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[42] Chi Zhang, Baoxiong Jia, Feng Gao, Yixin Zhu, Hongjing Lu, and Song-Chun Zhu. Learning perceptual inference by contrasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2

[43] Chi Zhang, Baoxiong Jia, Song-Chun Zhu, and Yixin Zhu. Abstract spatial-temporal reasoning via probabilistic abduction and execution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9736–9746, 2021. 2

[44] Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. Amr parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94, 2019. 6

[45] Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. Broad-coverage semantic parsing as transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3777–3789, 2019. 6

[46] Wenhe Zhang, Chi Zhang, Yixin Zhu, and Song-Chun Zhu. Machine number sense: A dataset of visual arithmetic problems for abstract and relational reasoning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020. 2