Complexité de l'échantillon et systèmes de questions réponses visuelles

Corentin Kervadec[1,2], Grigory Antipov[2], Moez Baccouche[2], Madiha Nadri[3], et Christian Wolf[1]

[1]Université de Lyon, INSA-Lyon, LIRIS, France.
[2]Orange, France
[3]Université de Lyon, Univ. Claude Bernard, LAGEPP, France.

Résumé

Les méthodes adressant la tâche des question-réponses visuelles, consistant à répondre à une question posée sur une image, sont connues pour leur tendance à exploiter les biais dans les données plutot que de raisonner. Récemment, il a été montré qu'il est possible de favoriser l'émergence de patterns de raisonnement dans les couches d'attention de modèles d'apprentissage profond de l'Etat-de-l'Art, en les entrainant sur des données visuelles parfaites. Ainsi, ces modèles sont capables de produire un raisonnement lorsque les conditions d'entrainement sont favorables. Cependant, il reste difficile de transferer ces parterns de raisonnement vers des modèles déployables, qui ne disposent pas d'une representation visuelle parfaite. Nous proposons une méthode de transfert basée sur un mécanisme de régularisation impliquant la supervision des séquences d'opérations nécéssaires pour répondre à la question. Nous donnons une analyse théorique basé sur le PAC-learning, montrant, sous conditions, qu'une telle supervision permet de réduire la compléxité de l'échantillon. Enfin, Nous démontrons experimentalement l'éfficacité de notre méthode sur la base de données GQA, ainsi que sa complémentarité avec les méthodes de pré-entrainement inspirées de BERT.

Abstract

Methods for Visual Question Anwering (VQA), which answer questions posed over images, are notorious for leveraging dataset biases rather than performing reasoning, hindering generalization. It has been recently shown that better reasoning patterns are emerging in attention layers of a state-of-the-art VQA model when they are trained on perfect (oracle) visual inputs, which provides evidence that deep neural networks can learn to reason when training conditions are favorable enough. However, transferring this learned knowledge to deployable models is a challenge, as knowledge is lost during the transfer. We propose a method for knowledge transfer based on regularization, which we design through an additional loss in the form of supervision of the sequence of required reasoning operations. We provide a theoretical analysis based on PAC-learning, showing that such program prediction can lead to decreased sample complexity under mild hypotheses. We also demonstrate the effectiveness of this approach experimentally on the GQA dataset and show its complementarity to BERT-like self-supervised pre-training.

1 Introduction

Reasoning over images is the main goal of Visual Question Anwering (VQA), a task where a model is asked to answer questions posed over images. This problem is a test bed for the creation of agents capable of high-level reasoning, as it involves multi-modal and high-dimensional data as well as complex decision functions involving latent representations and multiple hops. State-of-the-art models are notorious for leveraging dataset biases and short-cuts in learning rather than performing reasoning, leading to lack of generalization, as evidenced by extensive recent work on and bias oriented benchmarks for vision-and-language reasoning [TAH20, ABPK18, KABW21, KJA+21]. Even large-scale semi-supervised pre-training methods, which successfully managed to increase overall VQA performance, still struggle to address questions whose answers are rare given a context [KABW21].

It has been recently shown that reasoning patterns are emerging in attention layers of a SOTA VQA model when trained on perfect (oracle) visual inputs, which provide evidence that deep neural networks can learn to reason, when training conditions are favorable enough [KJA+21]. In particular, uncertainty and noise in

visual inputs seems to be a major cause for shortcut learning in VQA. While this kind of methods provides insights on the bottlenecks in problems involving learning to reason, and strong empirical results, they still suffer from significant loss in reasoning capabilities during the transfer phase, when the model is required to adapt from perfectly clean visual input to the noisy input it will encounter after deployment. We conjecture, that reasoning on noisy data involves additional components not necessary in the clean case due to different types of domain shifts :

— A *presence shift*, caused by imperfect object detectors, leads to missing visual objects necessary for reasoning, or to multiple (duplicate) detections.

— An *appearance shift* causes variations in object embeddings (descriptors) for the same class of objects due to different appearance.

In this paper, we propose a new method for transferring knowledge (reasoning patterns) from models learned on perfect visual input to models trained on noisy visual representations. Key to the success of the method is regularization minimizing loss of the reasoning capabilities during during transfer. In particular, we address this problem through program prediction as an additional loss, i.e. supervision of the sequence of reasoning operations along with their textual and/or visual arguments. The underlying hypothesis is additional grounding of the learning process through program supervision : in the knowledge transfer phase, when inputs are switched from clean oracle inputs to noisy input, the neural model is required continue to predict complex reasoning programs from different types of inputs, maintaining a strong link between the learned function and its objective.

As a second justification, we claim that program supervision in itself, i.e. w/o the context of knowledge transfer, leads to a simpler learning problem, as the underlying reasoning function is decomposed into a set of tasks, each of which is easier to learn the full joint decision function. We backup this claim through a theoretical analysis of sample complexity in reasoning in settings of additional supervision, showing decreased complexity under mild hypotheses.

In an experimental study, we demonstrate the effectiveness of this approach on the GQA dataset and show its complementarity even when combined to BERT-like self-supervised pre-training [TB19, DCLT19].

As a summary, this papers presents the following contributions :

— We propose a new program supervision module added on top of vision-language transformer models.

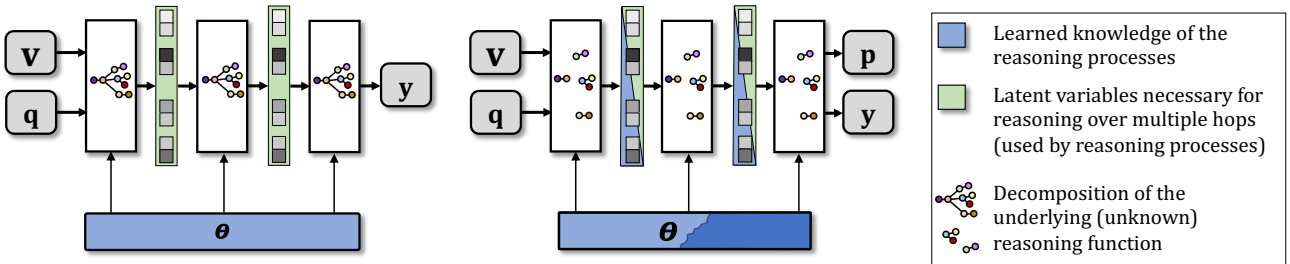
— We provide a theoretical analysis on the benefit of supervising program prediction in VQA deriving bounds on sample complexity.

— We experimentally demonstrate the efficiency of program supervision and show that (i) it increases VQA performance on both in- and out-of-distribution sets, even when combined with BERT-like pre-training [TB19, DCLT19] ; (ii) it improves the quality of oracle transfer initially proposed by [KJA+21].

2 Related Work

Visual Question Answering (VQA) — as a task was introduced in various datasets [AAL+15, GKSS+17, JHvdM+17], including GQA [HM19] which is automatically-generated from real-world images. Later, the GQA-OOD dataset [KABW20] introduced a new split of GQA focusing on rare (Out-Of-Distribution) question-answer pairs, and showed that many VQA models strongly rely on dataset biases. This growing amount of diverse datasets has been accompanied by the development of more sophisticated VQA models. While their exhaustive survey is out of the scope of this paper, one can mention families based on object-level attention [AHB+18] or tensor decomposition [BYCCT17]. In this work, we use Transformers [VSP+17] as VQA models due to their wide adoption by the community and impressive results. In particular, we focus on the combination of Transformers with a large-scale BERT [DCLT19]-like pretraining which was shown to be beneficial for VQA in several recent works [KABW19, TB19]. Our work has also connections with neural-symbolic reasoning [CGL+21, ARDK16] VQA models, which are generally based on the prediction of reasoning programs, whose elementary functions are learned jointly with program prediction itself. In contrast to this work, our method does not execute programs and does not use them for the prediction of answers.

Measuring complexity of learning problems — has been a goal of theoretical machine learning since the early days, with a large body of work based on PAC-Learning [Val84, SSSBD14]. Traditionally, bounds have been provided ignoring data distributions and focusing uniquely on hypothesis classes (network structures in neural network language), e.g. as measured by VC-dimension. Surprising experimental results on training networks on random samples have seemingly contradicted learning theory [ZBH+17], in



(a) Classical training, CE loss on answers **y** (b) CE loss on **y** + program prediction **p**

FIGURE 1 – VQA takes visual input $\boldsymbol{v}$ and a question $\boldsymbol{q}$ and predicts a distribution over answers $\boldsymbol{y}$. (a) Classical discriminative training results in the full reasoning function $g()$ being encoded in the network parameters θ, while the network activations contain latent variables necessary for reasoning over multiple hops (layers). (b) additional losses on program prediction require intermediate network activations to contain information on the reasoning process, simplifying learning the reasoning function g. Under the hypothesis of it's decomposition as $g() = \sum_r \boldsymbol{\pi}^{(r)}() g_r()$, intermediate supervision favors separately learning the reasoning mode selection $\boldsymbol{\pi}()$ and each reasoning mode function $g_r()$. This intuition is analyzed theoretically in section 3.

particular Rademacher Complexity. Recently, building on statistics of gradient descent, bounds have been proposed which take into account data distributions, notably [SAHLW19]. Algorithmic alignment between neural network structures and the decomposition of underlying reasoning functions has been studied in [XLZ+20], with a focus on algorithms based on dynamic programming. Our theoretical contribution in section 3 builds on the latter two methodologies and extends this type of analysis to intermediate supervision of reasoning programs.

3 Sample complexity of program supervision

We provide theoretical analysis showing that the prediction and supervision of reasoning programs can improve learnability (sample complexity) in vision and language reasoning under some assumptions. In what follows, we denote with g "true" (but unknown) underlying reasoning functions, and by f functions approximating them, implemented as neural networks. The goal is to learn a function g able to predict a distribution $\boldsymbol{y}$ over answer classes given an input question and an input image, see Fig 1a. While in the experimental part in section 4 we use state-of-the-art tranformer based models, in this theoretical analysis, we consider a simplified model, which takes as input the two vectorial embeddings $\boldsymbol{q}$ and $\boldsymbol{v}$ corresponding to, respectively, the question and the visual information (image), for instance generated by a language model and a convolutional neural network, and produces answers $\boldsymbol{y}^*$ as

$$\boldsymbol{y}^* = g(\boldsymbol{q}, \boldsymbol{v}). \tag{1}$$

We restrict this analysis to two-layer MLPs, as they are easier to handle theoretically than modern attention based models. The reasoning function g is approximated by neural network f parametrized by a vector θ and which predicts output answers $\boldsymbol{y}$ as

$$\boldsymbol{y} = f(\boldsymbol{q}, \boldsymbol{v}, \theta). \tag{2}$$

Our analysis uses PAC-learning [Val84] and builds on recent results providing bounds on sample complexity taking into account the data distribution itself. We here brievely recall and reproduce Theorem 3.5. from paper [XLZ+20], which, as an extension of a result in [SAHLW19], provides a bound for sample complexity of overparametrized MLPs with vectorial outputs, i.e. MLPs with sufficient capacity for learning a given task :

Theorem 3.1 (Sample complexity for overparametrized MLPs)**.** *Let $\mathcal{A}$ be an overparametrized and randomly initialized two-layer MLP trained with gradient descent for a sufficient number of iterations. Suppose $g : \mathbb{R}^d \to \mathbb{R}^m$ with components $g(x)^{(i)} = \sum_j \alpha_j^{(i)} (\beta_j^{(i)T} x)^{p_j^{(i)}}$, where $\beta_j^{(i)} \in \mathbb{R}^d$, $\alpha^{(i)} \in \mathbb{R}$, and $p_j^{(i)} = 1$ or $p_j^{(i)} = 2l, l \in \mathbb{N}_+$. The sample complexity $\mathcal{C}_{\mathcal{A}}(g, \epsilon, \delta)$ is*

$$\mathcal{C}_{\mathcal{A}}(g, \epsilon, \delta) = O\left(\frac{\max_i \sum_j p_j^{(i)} |\alpha_j^{(i)}| \cdot ||\beta_j^{(i)}||_2^{p_j^{(i)}} + \log(\frac{m}{\delta})}{(\epsilon/m)^2} \right) \tag{3}$$

We use the following *Ansatz* : since each possible input question requires a potentially different form of reasoning over the visual content, our analysis is based on the following assumption.

Assumption 1. *The unknown reasoning function $g()$ is a mixture model which decomposes as follows*

$$\boldsymbol{y}^* = \sum_r \boldsymbol{\pi}_r \boldsymbol{h}_r = \sum_r \boldsymbol{\pi}_r g_r(\boldsymbol{v}), \tag{4}$$

where the different mixture components r correspond to different forms of reasoning related to different questions. The mixture components can reason on the visual input only, and the mixture weights are determined by the question $\boldsymbol{q}$, i.e. the weights $\boldsymbol{\pi}$ depend on the question $\boldsymbol{q}$, e.g. $\boldsymbol{\pi} = g_\pi(\boldsymbol{q})$

We call $g_\pi(.)$ the *reasoning mode estimator*. One hypothesis underlying this analysis is that learning to predict reasoning programs allows the model to more easily decompose into this form (4), i.e. that the network structure closely mimicks this decomposition, as information on the different reasoning modes r is likely to be available in the activations of intermediate layers. This will be formalized in assumption 3 and justified further below.

Considering the supposed "true" reasoning function $\boldsymbol{y}^* = g(\boldsymbol{q}, \boldsymbol{v})$ and its decomposition given in (4), we suppose that each individual reasoning module g_r can be approximated with a multi-variate polynomial, in particular each component $\boldsymbol{h}_r^{(i)}$ of the vector $\boldsymbol{h}_r$, as

$$\boldsymbol{h}_r^{(i)} = g_r(\boldsymbol{v}) = \sum_j \alpha_{r,j}^{(i)} (\beta_{r,j}^{(i)T} \boldsymbol{v})^{p_{r,j}^{(i)}} \tag{5}$$

with parameters $\omega = \left\{ \alpha_{r,j}^{(i)}, \beta_{r,j}^{(i)}, p_{r,j}^{(i)} \right\}$.

A trivial lower bound on the complexity of the reasoning mode estimator $g_\pi(.)$ is the complexity of the identity function, which is trivially obtained in the highly unlikely case where the question embeddings $\boldsymbol{q}$ contain components corresponding to the 1-in-K encoding of the choice of reasoning mode r. We adopt a more realistic and typical case as the following assumption.

Assumption 2. *The input question embeddings $\boldsymbol{q}$ are separated into clusters according to reasoning modes r, such that underlying reasoning mode estimator g_π can be realized as a k-NN classifier with dot-product similarity in this embedding space.*

Under this assumption, the reasoning mode estimator can be expressed as a generalized linear model, i.e. a linear function followed by a soft-max σ,

$$\boldsymbol{\pi} = g_\pi(\boldsymbol{q}) = \sigma\left(\left[\gamma_0^T \boldsymbol{q},\ \gamma_1^T \boldsymbol{q}, ... \right] \right), \tag{6}$$

where the different γ_r are the cluster centers of the different reasoning modes r in the question embedding space. As the softmax is a monotonic non-linear function, its removal will not decrease sample complexity [1], and the complexity can be bounded by the logits $\boldsymbol{\pi}_r = \gamma_r^T \boldsymbol{q}$. Plugging this into (4) we obtain that each component $\boldsymbol{y}^{*(i)}$ of the answer is expressed as the following function :

$$\boldsymbol{y}^{*(i)} = \sum_r \left(\gamma_r^T \boldsymbol{q} \right) \sum_j \alpha_{r,j}^{(i)} (\beta_{r,j}^{(i)T} \boldsymbol{v})^{p_{r,j}^{(i)}} \tag{7}$$

We can reparametrize this function by concatenating the question $\boldsymbol{q}$ and the visual input $\boldsymbol{v}$ into a single input vector $\boldsymbol{x}$, which are then masked by two different binary masks, which can be subsumed into the parameters γ_r and $\beta_{r,j}^{(i)}$, respectively, giving

$$\boldsymbol{y}^{*(i)} = \sum_r \sum_j (\gamma_r^T \boldsymbol{x}) \alpha_{r,j}^{(i)} (\beta_{r,j}^{(i)T} \boldsymbol{x})^{p_{r,j}^{(i)}} \tag{8}$$

Extending Theorem 3.5. from [XLZ+20], we can give our main theoretical result as the sample complexity of this function expressed as the following theorem.

Theorem 3.2 (Sample complexity for multi-mode reasoning functions)**.** *Let $\mathcal{A}$ be an overparametrized and randomly initialized two-layer MLP trained with gradient descent for a sufficient number of iterations. Suppose $g : \mathbb{R}^d \to \mathbb{R}^m$ with components $g(x)^{(i)} = \sum_r \sum_j (\gamma_r^T \boldsymbol{x}) \alpha_{r,j}^{(i)} (\beta_{r,j}^{(i)T} \boldsymbol{x})^{p_{r,j}^{(i)}}$ where $\gamma_r \in \mathbb{R}^d$, $\beta_{r,j}^{(i)} \in \mathbb{R}^d$, $\alpha_{r,j}^{(i)} \in \mathbb{R}$, and $p_{r,j}^{(i)} = 1$ or $p_{r,j}^{(i)} = 2l, l \in \mathbb{N}_+$. The sample complexity $\mathcal{C}_\mathcal{A}(g, \epsilon, \delta)$ is*

$$\mathcal{C}_\mathcal{A}(g, \epsilon, \delta) = O\left(\frac{\max_i \sum_r \sum_j \pi p_{r,j}^{(i)} |\alpha| \cdot ||\gamma_r||_2 \cdot ||\beta_{r,j}||_2^{p_{r,j}^{(i)}} + \log(m/\delta)}{(\epsilon/m)^2} \right),$$

The proof of this theorem is given in the supplementary material (Appendix A).

Theorem 3.2 provides the sample complexity of the reasoning function $g()$ under classical training. In the case of program supervision, our analysis is based on the following assumption (see also Fig. 1b) :

Assumption 3. *Supervising the prediction of reasoning programs encodes the choice of reasoning modes r into the hidden activations of the network f. Therefore, learning is separated into several different processes,*

1. In principle, there should exist special degenerate cases, where an additional softmax could reduce sample complexity ; however, in our case it is applied to a linear function and thus generates a non-linear function.

(a) learning of the reasoning mode estimator $g_\pi()$ approximated as a network branch $f_\pi()$ connected to the program output ;

(b) learning of the the different reasoning modules $g_r()$ approximated as network branches $f_r()$ connected to the different answer classes $\boldsymbol{y}_r$; each one of these modules is learned independently.

We justify Assumption 3.a through supervision directly, which separates $g_\pi()$ from the rest of the reasoning process. We justify Assumption 3.b by the fact, that different reasoning modes r will lead to different hidden activations of the network. Later layers will therefore see different inputs for different modes r, and selector neurons can identify responsible inputs for each branch $f_r()$, effectively switching off irrelevant input.

We can see that these complexities are lower than the sample complexity of the full reasoning function given in theorem 3.2, since for a given combination of i, r, j, the term $||\gamma_r||_2 \cdot ||\beta_{r,j}||_2^{p_{r,j}^{(i)}}$ dominates the corresponding term $||\beta_{r,j}||_2^{p_{r,j}^{(i)}}$. Let us recall that the different vectors γ correspond to the cluster centers of reasoning modes in language embedding space. Under the assumption that the language embeddings $\boldsymbol{q}$ have been created with batch normalization, a standard technique in neural vision and language models, each value $\gamma_r^{(i)}$ follows a normal distribution $\mathcal{N}(0,1)$. Dropping indices i, r, j to ease notation, We can then compare the expectation of the term $||\gamma||_2 \cdot ||\beta||_2^p$ over the distribution of γ and derive the following relationship :

$$\mathbb{E}_{\gamma^{(i)} \sim N(0,1)} ||\gamma||_2 \cdot ||\beta||_2^p \;=\; C||\beta||_2^p \;=\; \sqrt{2}\frac{\Gamma(\frac{m}{2}+\frac{1}{2})}{\Gamma(\frac{m}{2})}||\beta||_2^p \tag{9}$$

where Γ is the Gamma special function and m is the dimension of the language embedding γ. We provide a proof for this equality in the supplementary material (Appendix B).

Discussion and validity of our claims — the difference in sample complexity is determined by the factor C in equation (9), which monotonically grows with the size of the embedding space m, which is typically in the hundreds. For the order of m=512 to m=768 used for state-of-the-art LXMERT models [TB19], complexity grows by a factor of around $\sim$20.

We would like to point out, that this analysis very probably under-estimates the difference in complexity, as the difference very much depends on the complexity of the reasoning estimator $\boldsymbol{\pi}$, which we have simplified as a linear function in equation (6). Taking into account just the necessary soft-max alone would probably better appreciate the difference in complexity between the two methods, which we leave for future work.

Our analysis is also based on several assumptions, among which is the simplified model (an over-parametrized MLP instead of an attention based network), as well as assumptions of Theorem 3.2 from [XLZ+20] and [SAHLW19], on which our analysis is based.

Lastly, we would like to comment on the fact that we compare two different bounds : (i) the bound on sample complexity for learning the full multi-modal reasoning given in Theorem 3.2, and (ii) the bound for learning a single reasoning mode given by Theorem 3.1. While comparing bounds does not provide definitive answers on the order of models, both bounds have been derived by the same algebraic manipulations and we claim that they are comparable.

4 Knowledge transfer and program supervision

We now experimentally demonstrate the effectiveness of our approach by augmenting a State-Of-the-Art VQA model with program supervision. For this purpose, we use the vision and language transformer model LXMERT [TB19] based on sequences of self-attention and cross-modality attention. We add the dedicated module for program generation to the output of the base model as shown in Fig. 2 — an adaptation to other architectures would be straight forward. In the lines of [CGL+21], the program decoder has been designed in a coarse-to-fine fashion. It first generates a coarse sketch of the program consisting only of the main operations, which are then refined by predicting dependencies, textual and visual arguments.

Coarse : operation — coarse prediction only predicts the main reasoning steps, i.e. a sequence of embeddings corresponding to the operations ("select color", "choose size", "query name", "and", etc.). This sequence $\{O_i\}_{i\in[0,n-1]} \in \mathcal{R}^{d_h}$ is inferred using a recurrent neural network (RNN) in a GRU [CvMG+14] variant. The GRU's input is pooled from the base model's output, in particular the task-specific y_{CLS} token embedding.

Fine : arguments — the coarse program is then refined by adding the operations' arguments to the prediction. We define two types of arguments : the question words and the image regions, both as choices over the input tokens to VL-transformer model. We associate the arguments to each operation by computing an affinity score s_{ij} between each operation embedding

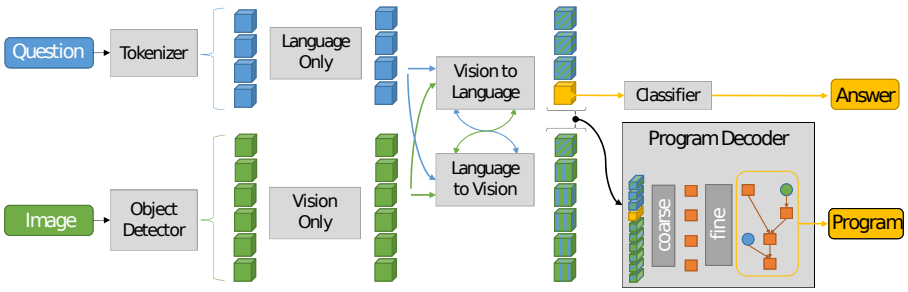


FIGURE 2 – Program supervision. Placement of the program decoder into an LXMERT vision-language transformer. The decoder is fed with LXMERT outputs (enriched language and vision embeddings) and generates programs using a coarse-to-fine approach.

O_i and each token embedding v_j or l_j, which represents the probability for word l_j or v_j to belong to the argument set of operation O_i. It is computed with a 2-layer feed-forward network taken concatenated input vectors.

Graph structure — While the program is predicted as a sequence, it is actually structured as a graph (a tree). In the question "*Is there a motorbike or a plane ?*", for instance, the operation "or" depends on the result of two operations checking the existence of a specific object in the image. We use another RNN on top of the operation embeddings O_i to predict the indices of the parent functions.

Program supervision — The corse-to-fine program decoder trained using four additional losses :

$$\mathcal{L} = \underbrace{\mathcal{L}_{vqa}}_{\text{VQA}} + \underbrace{\alpha.\mathcal{L}_{op} + \beta.\mathcal{L}_{dep} + \gamma.\mathcal{L}_{qarg} + \delta.\mathcal{L}_{varg}}_{\text{Program supervision}}, \quad (10)$$

where $\mathcal{L}_{op}$ is a cross-entropy loss measuring prediction error over operations. Argument losses $\mathcal{L}_{qarg}$ and $\mathcal{L}_{varg}$ (resp. textual and visual) are measured with a binary cross entropy loss. Dependencies between operations are supervised with another cross-entropy loss $\mathcal{L}_{dep}$. α, β, γ, δ are hyper-parameters weighting the different losses.

Ground truth programs — We use ground truth information from the GQA [HM19] dataset, whose questions have been automatically generated from real images. It contains a program describing the operations and arguments required to find the answer for each question. However, the visual ground-truth arguments (image regions) do not exactly match the visual model's input, as visual input is obtained from an object detector [AHB+18]. We therefore match GT visual argument labels to detected objects minimizing intersection-over-union (IoU).

Oracle transfer — Our method uses program supervision to regularize knowledge transfer from a visual oracle to noisy input, as introduced in [KJA+21]. Oracle transfer consists in pretraining the VL-Transformer model on ground-truth visual input before performing BERT-like pre-training, it offers training conditions which are more favorable for learning reasoning capabilities. We perform the following steps :

- Oracle pre-training on GT visual input on the GQA dataset, including program supervision.
- (optionally) BERT-like pre-training on data from GQA + MS COCO, with program-supervision only on the samples taken from GQA.
- Finetuning on the final VQA-objective on the GQA dataset, while keeping program supervision.

5 Experimental results

Datasets and experimental setup — We perform our experiments with a tiny version of LXMERT [TB19], characterized by a hidden embedding size of d=128 and h=4 heads per layer. We evaluate on the GQA [HM19] and GQA-OOD [KABW20] datasets. GQA-OOD is a benchmark dedicated to the out-of-domain VQA evaluation, and gives information on the generalisation power of the model. GQA is particularly well suited for evaluating a large variety of reasoning skills.

Training details — All models were trained with the Adam optimizer [KB14], a learning rate of 10^{-4} with warm starting and learning rate decay, on two P100 GPUs. BERT/LXMERT [TB19] pretraining is performed during 20 epochs with a batch size of 512. All pretraining losses are added from the beginning, including the VQA one. Note that LXMERT [TB19] is originally pre-trained on a corpus gathering images

Model	Pretraining		Programs	GQA [HM19]
	Oracle	LXMERT/BERT		overall acc.
(a) Baseline (+LXMERT/BERT)		✓		56.8
(b) Oracle transfer (+LXMERT/BERT)	✓	✓		57.8
(c) Ours (+LXMERT/BERT)	✓	✓	✓	58.4

TABLE 1 – Impact of program supervision on vision-language transformers ; All models are pre-trained with LXMERT [TB19]/BERT-like objectives after *Oracle Transfer*. We report scores on GQA [HM19]-val, hyper-parameters selected on validation set.

Supervision of	GQA-OOD [KABW20]		GQA [HM19]
	acc-tail	acc-head	overall
(0) VQA only	46.9	62.0	62.2
(a) Coarse only	46.5	63.9	62.5
(b) Coarse + dependencies	46.8	64.4	62.8
(c) Full w/o v.arg	47.3	65.1	63.7
(d) **Full (ours)**	**49.9**	**67.8**	**66.2**

TABLE 2 – Ablation study of different types of program supervision (tiny model, no LXMERT/BERT pre-training), on GQA validation. *v.arg* = supervision of visual arguments.

and sentences from MSCOCO [LMB+14] and Visual-Genome [KZG+17]. As the GQA dataset is built upon VisualGenome, the original LXMERT pre-training dataset contains samples from the GQA validation split. Therefore, *we removed these validation samples from the pre-training corpus*, in order to be able to validate on the GQA validation split. After pre-trainning, we finetune on GQA [HM19] during 4 epochs, with a batch size of 32 and a learning rate equal to 10^{-4}. Program decoder hyper-parameters are set to $\alpha = 1$, $\beta = 1$, $\gamma = 1$ and $\delta = 100$.

Program supervision improves visual reasoning — Table 1 reports the effectiveness of program prediction when combined with oracle and BERT-like pretraining on the GQA dataset and experimentally confirms the results found in the theoretical analysis.

Visual arguments are the key — We study the impact of different types of program supervision in Table 2. We can see that the importance of supervising arguments, in (c) and (d). The supervision of visual arguments (d) contributes most to the gain in performance, again corroborating that visual uncertainty is the main bottleneck for reasoning on the GQA dataset. In addition, results on GQA-OOD (*acc-tail* and *acc-head*) suggest that the gains are obtained in, both, out- and in-distribution settings.

6 Conclusion

We have demonstrated that it is possible to improve the reasoning abilities of VQA models when providing additional supervision of program annotations, and we have shown that this achieves superior performances on OOD settings. In a theoretical analysis, we have shown that program supervision can decrease sample complexity under reasonable hypothesis.

The proposed method relies on the availability of reasoning program annotations, which are costly to annotate, especially when dealing with human generated questions. Recent work has already managed to gather such kind of annotations [DAZ+16]. The next step will be to extend the method to configurations where the program annotation is rare or incomplete.

Références

[AAL+15] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa : Visual question answering. In *ICCV*, 2015.

[ABPK18] Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. Don't just assume ; look and answer : Overcoming priors for visual question answering. In *CVPR*, 2018.

[AHB+18] 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 *CVPR*, pages 6077–6086, 2018.

[ARDK16] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *CVPR*, 2016.

[BYCCT17] Hedi Ben-Younes, Rémi Cadene, Matthieu Cord, and Nicolas Thome. Mutan : Multimodal tucker fusion for visual question answering. In *ICCV*, 2017.

[CGL+21] Wenhu Chen, Zhe Gan, Linjie Li, Yu Cheng, William Wang, and Jingjing Liu. Meta module network for compositional visual reasoning. In *WACV*, 2021.

[CvMG+14] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.

[DAZ+16] Abhishek Das, Harsh Agrawal, C. Lawrence Zitnick, Devi Parikh, and Dhruv Batra. Human Attention in Visual Question Answering : Do Humans and Deep Networks Look at the Same Regions ? In *EMNLP*, 2016.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert : Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, 2019.

[GKSS+17] 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 *CVPR*, pages 6904–6913, 2017.

[HM19] Drew A Hudson and Christopher D Manning. Gqa : A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, pages 6700–6709, 2019.

[JHvdM+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 *CVPR*, pages 2901–2910, 2017.

[KABW19] Corentin Kervadec, Grigory Antipov, Moez Baccouche, and Christian Wolf. Weak supervision helps emergence of word-object alignment and improves vision-language tasks. In *European Conference on Artificial Intelligence*, 2019.

[KABW20] C. Kervadec, G. Antipov, M. Baccouche, and C. Wolf. Roses Are Red, Violets Are Blue... but Should VQA Expect Them To ? *Pre-print : arxiv :2006.05121*, 2020.

[KABW21] Corentin Kervadec, Grigory Antipov, Moez Baccouche, and Christian Wolf. Roses are red, violets are blue... but should vqa expect them to ? In *CVPR*, 2021.

[KB14] Diederik P Kingma and Jimmy Ba. Adam : A method for stochastic optimization. 2014.

[KJA+21] Corentin Kervadec, Theo Jaunet, Grigory Antipov, Moez Baccouche, Romain Vuillemot, and Christian Wolf. How transferable are reasoning patterns in vqa ? In *CVPR*, 2021.

[KZG+17] 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. *IJCV*, 123(1) :32–73, 2017.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco : Common objects in context. In *ECCV*, 2014.

[SAHLW19] S.S. Du S. Arora, W. Hu, Z. Li, and R. Wang. Fine-grained Analysis of optimization and generalization for overparametrized two-layer neural networks. In *ICML*, 2019.

[SSSBD14] Shai S. Shalev-Shwart and S. Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014.

[TAH20] Damien Teney, Ehsan Abbasnejad, and Anton van den Hengel. Unshuffling data for improved generalization. *arXiv preprint arXiv :2002.11894*, 2020.

[TB19] Hao Tan and Mohit Bansal. Lxmert : Learning cross-modality encoder representations from transformers. In *EMNLP*, pages 5103–5114, 2019.

[Val84] L.G. Valiant. A theory of the learnable. In *Communications of the ACM*, volume 27(11), 1984.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[XLZ+20] K. Xu, J. Li, M. Zhang, S.S. Du, K.-I. K., and S. Jegelka. What can Neural Networks Reason About. In *ICLR*, 2020.

[ZBH+17] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization . In *ICLR*, 2017.

Supplementary Material

A Proof of theorem 3.2

For the unfamiliar reader, we here briefly recall the notion of sample complexity, in the context of PAC-learning [Val84], which characterizes the minimum amount (=M) of samples necessary to learn a function with sufficiently low (= ϵ) error with sufficiently high (= δ) probability :

Definition A.1 (Sample complexity)**.** Given an error threshold $\epsilon>0$; a threshold on error probability δ; a training set $S = \{x_i, y_i\}$ of M i.i.d. training samples from $\mathcal{D}$, generated from some underlying true function $\underline{y}_i = g(\boldsymbol{x}_i)$, and a learning algorithm $\mathcal{A}$, which generates a function f from training data, e.g. $f = \mathcal{A}(S)$; Then g is (M, ϵ, δ)-learnable by $\mathcal{A}$ if

$$\mathbb{P}_{x\sim D}\left[||f(x) - g(x)|| \leq \epsilon\right] \geq 1 - \delta \tag{11}$$

A.1 The case of scalar outputs

In the lines of [SAHLW19], we first define the case for a single component $\boldsymbol{y}^{(i)}$ of the vector $\boldsymbol{y}$ and define the following Corollary :

Corollary 0.1 (Sample complexity for multi-mode reasoning functions with a single scalar component)**.** *Let $\mathcal{A}$ be an overparametrized and randomly initialized two-layer MLP trained with gradient descent for a sufficient number of iterations. Suppose $g : \mathbb{R}^d \to \mathbb{R}^m$ with $g(x) = \sum_r \sum_j (\gamma_r^T \boldsymbol{x})\alpha_{r,j}(\beta_{r,j}^T \boldsymbol{x})^{p_{r,j}}$ where $\gamma_r \in \mathbb{R}^d$, $\beta_{r,j} \in \mathbb{R}^d$, $\alpha_{r,j} \in \mathbb{R}$, and $p_{r,j} = 1$ or $p_{r,j} = 2l, l \in \mathbb{N}_+$. The sample complexity $\mathcal{C}_{\mathcal{A}}(g, \epsilon, \delta)$ is*

$$C_A(g, \epsilon_0, \delta_0) = O\left(\frac{\sum_r \sum_j \pi p_{r,j} |\alpha| \cdot ||\gamma_r||_2 \cdot ||\beta_{r,j}||_2^{p_{r,j}} + \log(\frac{1}{\delta_0})}{\epsilon_0^2}\right),$$

Proof of Corollary 0.1 :

Using Theorem 5.1 from [SAHLW19], we know that sums of learnable functions are learnable, and can thus focus on a single term

$$y = g(\boldsymbol{x}) = \alpha(\gamma^T \boldsymbol{x})(\beta^T \boldsymbol{x})^p \tag{12}$$

where we dropped indices r and j and the superscript (i) for convenience.

We proceed in the lines of the proof of Theorem 5.1 in [SAHLW19]. Given a set of i.i.d data samples $S = \{(\boldsymbol{x}_s, y_s)\}_{s=1}^n = (\boldsymbol{X}, \boldsymbol{y})$ from the underlying function $g(x)$, let $\boldsymbol{w}$ be the weights of the first layer of two layer network with ReLu activations; let $\boldsymbol{H}^\infty \in \mathbb{R}^{n,n}$ be a Gram matrix defined as follows, with elements

$$\boldsymbol{H}_{ij}^\infty = \mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(0,1)}\left[\boldsymbol{x}_i^T \boldsymbol{x}_j \mathbb{I}\{\boldsymbol{w}^t \boldsymbol{x}_i \geq 0, \boldsymbol{w}^t \boldsymbol{x}_i \geq 0\}\right].$$

To provide bounds on the sample complexity of $g(x)$, using Theorem 5.1 of [SAHLW19], it suffices to show that the following bound holds

$$\sqrt{\boldsymbol{y}^T (\boldsymbol{H}^\infty)^{-1} \boldsymbol{y}} < M_g \tag{13}$$

for a bound M_g independent of the number of samples n.

For first introduce some notation. For matrices $\boldsymbol{A} = [\boldsymbol{a}_1, ..., \boldsymbol{a}_{n_3}] \in \mathbb{R}^{n_1 \times n_3}$ and $\boldsymbol{B} = [\boldsymbol{b}_1, ..., \boldsymbol{b}_{n_3}] \in \mathbb{R}^{n_2 \times n_3}$, the *Khatri-Rao* product is defined as $\boldsymbol{A} \odot \boldsymbol{B} = [\boldsymbol{a}_1 \otimes \boldsymbol{b}_1, \boldsymbol{a}_2 \otimes \boldsymbol{b}_2, ..., \boldsymbol{a}_{n_3} \otimes \boldsymbol{b}_{n_3}]$. Let $\circ$ be the *Haddamard* product (element wise multiplication) of two matrices. We also denote the corresponding powers by $\boldsymbol{A}^{\otimes l}, \boldsymbol{A}^{\odot l}, \boldsymbol{A}^{\circ l}$. We denote by $\boldsymbol{A}^\dagger = (\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}^T$ the *Moore-Penrose* pseudo-inverse, and by $\boldsymbol{P}_{\boldsymbol{A}} = \boldsymbol{A}^{\frac{1}{2}} \boldsymbol{A}^\dagger \boldsymbol{A}^{\frac{1}{2}}$ the projection matrix for the subspace spanned by $\boldsymbol{A}$.

From the proof of Theorem 5.1 in [SAHLW19], we also know that

$$\boldsymbol{H}^\infty \succeq \frac{\boldsymbol{K}^{\circ 2l}}{2\pi(2l-1)^2},$$

where $\boldsymbol{K} = \boldsymbol{X}^T \boldsymbol{X}$, and $\boldsymbol{X}$ is the data matrix of all row vectors $\boldsymbol{x}_i$.

Let us consider the case of $p = 1$. Reformulating equation (12), we get :

$$y = g(\boldsymbol{x}) = \alpha(\gamma^T \boldsymbol{x})(\beta^T \boldsymbol{x}) \tag{14}$$
$$= \alpha(\boldsymbol{x}^T \gamma)(\boldsymbol{x}^T \beta) \tag{15}$$
$$= \alpha(\boldsymbol{x} \otimes \boldsymbol{x})^T (\gamma \otimes \beta) \tag{16}$$

Now, taking the full set of input vectors $\boldsymbol{x}_i$ arranged into the full data matrix $\boldsymbol{X}$, we can perform similar algebraic operations to get

$$\boldsymbol{y} = g(\boldsymbol{X}) = \alpha(\boldsymbol{X}^T \gamma) \circ (\boldsymbol{X}^T \beta) \tag{17}$$
$$= \alpha(\boldsymbol{X}^{\odot 2})^T (\gamma \otimes \beta) \tag{18}$$

Plugging (17) and (18) into (13), we need to show that

the following expression is smaller than a constant M_g :

$$\alpha^2((\boldsymbol{X}^T\gamma)\circ(\boldsymbol{X}^T\beta))^T(\boldsymbol{H}^\infty)^{-1}(\boldsymbol{X}^{\odot 2})^T(\gamma\otimes\beta) \quad (19)$$
$$=\alpha^2((\boldsymbol{X}^{\odot 2})^T(\gamma\otimes\beta))^T(\boldsymbol{H}^\infty)^{-1}(\boldsymbol{X}^{\odot 2})^T(\gamma\otimes\beta) \quad (20)$$
$$=\alpha^2(\gamma\otimes\beta)^T(\boldsymbol{X}^{\odot 2})(\boldsymbol{H}^\infty)^{-1}(\boldsymbol{X}^{\odot 2})^T(\gamma\otimes\beta) \quad (21)$$
$$\leq 2\pi\alpha^2(\gamma\otimes\beta)^T(\boldsymbol{X}^{\odot 2})(\boldsymbol{K}^{\circ 2})^\dagger(\boldsymbol{X}^{\odot 2})^T(\gamma\otimes\beta) \quad (22)$$
$$=2\pi\alpha^2(\gamma\otimes\beta)^T\boldsymbol{P}_{\boldsymbol{X}^{\odot 2}(\boldsymbol{X}^{\odot 2})^T}(\gamma\otimes\beta) \quad (23)$$
$$\leq 2\pi\alpha^2||(\gamma\otimes\beta)||_2^2 \quad (24)$$
$$=2\pi\alpha^2||\gamma||_2^2\cdot||\beta||_2^2 \quad (25)$$

where we made use of $||a\otimes b||_2^2 = ||a||_2^2||b||_2^2$ for two vectors a and b and an integer n.

This finishes the proof for the case $p = 1$.

Let us consider the case of $p = 2l+1$. Reformulating equation (12), we get :

$$\boldsymbol{y} = g(\boldsymbol{X}) = \alpha(\boldsymbol{X}^T\gamma)\circ(\boldsymbol{X}^T\beta)^p \quad (26)$$
$$= \alpha(\boldsymbol{X}^{\odot 2l})^T(\gamma\otimes\beta^{\otimes(2l+1)}) \quad (27)$$

Plugging (27) into (13), we again need to show that the following expression is smaller than a constant M_g :

$$\alpha^2((\boldsymbol{X}^{\odot 2l})^T(\gamma\otimes\beta^{\otimes(2l+1)}))^T \quad (28)$$
$$(\boldsymbol{H}^\infty)^{-1}(\boldsymbol{X}^{\odot 2l})^T(\gamma\otimes\beta^{\otimes(2l+1)}) \quad (29)$$
$$=\alpha^2(\gamma\otimes\beta^{\otimes(2l+1)})^T \quad (30)$$
$$(\boldsymbol{X}^{\odot 2l})(\boldsymbol{H}^\infty)^{-1}(\boldsymbol{X}^{\odot 2l})^T(\gamma\otimes\beta^{\otimes(2l+1)}) \quad (31)$$
$$\leq 2\pi(2l-1)^2\alpha^2(\gamma\otimes\beta^{\otimes(2l+1)})^T \quad (32)$$
$$(\boldsymbol{X}^{\odot 2l})(\boldsymbol{K}^{\circ 2})^\dagger(\boldsymbol{X}^{\odot 2l})^T(\gamma\otimes\beta^{\otimes(2l+1)}) \quad (33)$$
$$=2\pi(2l-1)^2\alpha^2(\gamma\otimes\beta^{\otimes(2l+1)})^T \quad (34)$$
$$\boldsymbol{P}_{\boldsymbol{X}^{\odot 2l}(\boldsymbol{X}^{\odot 2l})^T}(\gamma\otimes\beta^{\otimes(2l+1)}) \quad (35)$$
$$\leq 2\pi(2l-1)^2\alpha^2||(\gamma\otimes\beta^{\otimes(2l+1)})||_2^2 \quad (36)$$
$$\leq 2\pi p^2\alpha^2||(\gamma\otimes\beta^{\otimes(2l+1)})||_2^2 \quad (37)$$
$$=2\pi p^2\alpha^2||\gamma||_2^2\cdot||\beta||_2^{2p} \quad (38)$$

where we made use of $||a\otimes b||_2^2 = ||a||_2^2||b||_2^2$ and therefore $||a^{\otimes n}||_2^2 = ||a||_2^{2n}$ for two vectors a and b and an integer n.

This finishes the proof for the case $p = 2l+1$.

A.2 The case of vectorial outputs

In the lines of [XLZ+20], we consider each component of the output vector independent and apply an union bound to Corollary 0.1. If the individual components $\boldsymbol{y}^{(i)}$ fail to learn with probability δ_0, then the full output of dimension m fails with probability $m\delta_0$ and with an error of at most $m\epsilon_0$. A change of variables from (ϵ_0, δ_0) to (ϵ, δ) gives a complexity for the model with vectorial output of

$$\mathcal{C}_{\mathcal{A}}(g,\epsilon,\delta) = O\left(\frac{\max_i\sum_r\sum_j\pi p_{r,j}^{(i)}|\alpha|\cdot||\gamma||_2\cdot||\beta_{r,j}||_2^{p_{r,j}^{(i)}}+\log(m/\delta)}{(\epsilon/m)^2}\right),$$

This ends the proof of Theorem 3.2.

B Proof of the inequality in Eq. (9)

Let us denote by $p(x)$ the density of normal distribution. And to make the notation more succinct and to avoid confusion between different usages of superscripts, in this proof we will change γ_r^i to γ_i, i.e. the i^{th} component of the vector γ, not to be confused with γ_r, a vector corresponding to the embedding of the r^{th} reasoning mode. Then,

$$\mathbb{E}_{\gamma_i\sim N(0,1)}||\gamma||_2\cdot||\beta||_2^p \quad (39)$$
$$=||\beta||_2^p\mathbb{E}_{\gamma_i\sim N(0,1)}\left(\sum_i\gamma_i^2\right)^{\frac{1}{2}} \quad (40)$$
$$(41)$$

We now perform a change of variables and introduce a new random variable

$$z = \sum_i\gamma_i^2. \quad (42)$$

Since each individual γ_i is distributed normal, z is distributed according to a χ^2 distribution with m degrees of freedom, and we get

$$\mathbb{E}_{\gamma_i\sim N(0,1)}||\gamma||_2\cdot||\beta||_2^p \quad (43)$$
$$=||\beta||_2^p\ \mathbb{E}_{z\sim\chi^2}[z^{\frac{1}{2}}] \quad (44)$$

The expectation now corresponds to $\frac{1}{2}^{th}$ centered moment of the χ^2 distribution with m degrees of freedom, whose k^{th} moments are given as

$$\mathbb{E}_{z\sim\chi^2}[z^k] = 2^k\frac{\Gamma(\frac{m}{2}+k)}{\Gamma(\frac{m}{2})} \quad (45)$$

This ends the proof of the equality.