Neural Module Networks for Compositional Visual Reasoning

PhD Thesis Defense of M^{me} Wafa AISSA

Jury members:

M. Camille KURTZ	Professeur, LIPADE, Univ. Paris Cité	Rapporteur
M. Désiré SIDIBE	Professeur, IBISC, Univ. d'Evry	Rapporteur
M^{me} Valérie GOUET-BRUNET	Directrice de recherche, LaSTIG, IGN	Examinatrice
M^{me} Anissa MOKRAOUI	Professeur, L2TI, Univ. Sorbonne Paris Nord	Examinatrice
M. Michel CRUCIANU	Professeur, CEDRIC, Cnam	Directeur de thèse
M. Marin FERECATU	Maître de conférences, CEDRIC, Cnam	Co-encadrant
M. Souheil HANOUNE	Chief technology officer, XXII Group	Co-encadrant

Content

Introduction

Visual Reasoning

- Visual reasoning = To reason about the visual world = To manipulate previously acquired knowledge to **understand** an image in order to make a decision.
- Visual reasoning goes beyond mere visual recognition.
- Evaluation benchmark: Visual Question Answering (VQA).
- VQA approaches: monolithic and **compositional**.

Motivations

- Visual reasoning about **complex scenes** is extremely hard.
- **Multi-modality**: VQA merges CV with NLP.
- **Explicit reasoning** through compositionality.

Compositional Visual Reasoning for VQA

- Visual reasoning is inherently **compositional**.
- Break down the question into **modular** sub-tasks.
- Assign each **sub-task** to a different module.
- **Reasoning skills**: object and attribute detection, relation extraction, counting, comparisons, logic operations ...

Figure 1: Q1: What color is the fruit on the right side, red or green?
Q2: Is there any milk in the bowl to the left of the apple?

Related work

Visual Question Answering (VQA) architectures

Monolithic

- Static architecture using fusion mechanisms to connect vision and language.
- Examples: LXMERT [34], ViLBERT [28], VisualBERT [25].

Compositional

- Complex problems are composed of multiple smaller, interconnected sub-problems.

Multi-step

- Recurrent architecture.
- Implicit reasoning guided by the question attention.
- Examples: SAN [37], FiLM [30], MAC [17].

Neural Module Networks (NMN)

- Dynamic architecture adapted to the question.
- Explicit reasoning guided by a reasoning chain.
- Examples: NMN [5], InfExec [20], PVR [24], MMN [11].

VQA monolithic: generalities

- CNN/Transformer extract image features.
- LSTM/Transformer extract question embeddings.
- Multi-modal attention and fusion.

+ Performance gains due to the power of DNNs.

– Lack explainability.

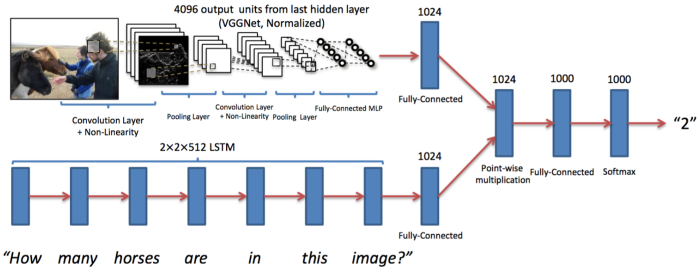


Figure 2: Model from [6].

- VLP model trained on various unimodal and multi-modal tasks.
- Bottom-up features: region-based features extracted by Faster-RCNN object detector [31].
- BERT-like encoder blocks [12].
  - Self-attention for intra-modal relationships. $\text{SelfAtt}_{L \to L}$, $\text{SelfAtt}_{V \to V}$.
  - Cross-attention for inter-modal alignment. $\text{CrossAtt}_{L \to V}$, $\text{CrossAtt}_{V \to L}$.

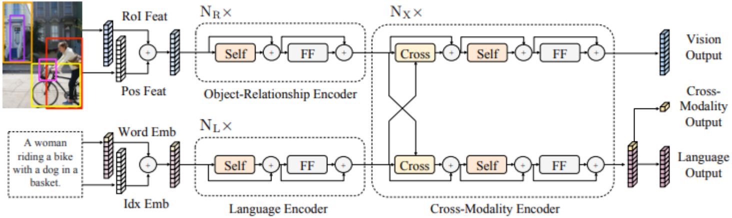


Figure 3: LXMERT: Learning Cross-Modality Encoder Representations from Transformers [34].

- Modular, composable and **jointly trained** NNs framework for VQA.
- Decompose questions into linguistic substructures using NL parser [23].
- Each module operates to accomplish a different sub-task.
- **Transparency** and **interpretability**.

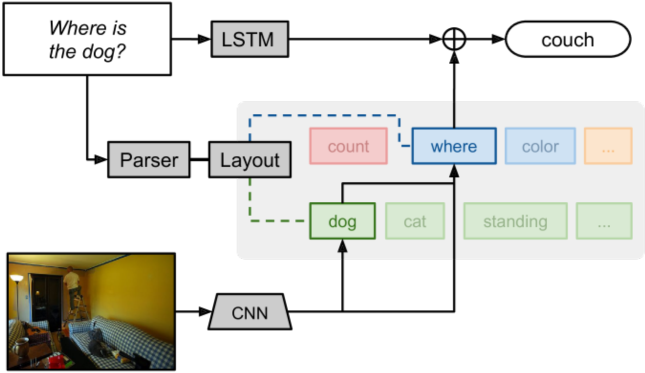


Figure 4: NMN example [5], layout= `classify[where](attend[dog])`

Neural Module networks (NMN): Overview

Modules taxonomy

- Set of predefined modules representing functions.
  - Examples: Select, Filter, Relate, And, Or, Query ...

Generator

- Transforms a natural language question into a program format.
  - Example: Where is the dog? → `Select(dog), QueryPos()`.

Executor

- Instantiates the program into an NMN.
- Executes it on the image.
- Predicts an answer.

- A more end-to-end approach.
- **Machine translation**: LSTM to generate the Programs.
- **Generic modules**: modules use arguments: $\cancel{\text{find}_1[\text{circle}], \text{find}_2[\text{cube}]}$ $\Rightarrow$ find("argument")

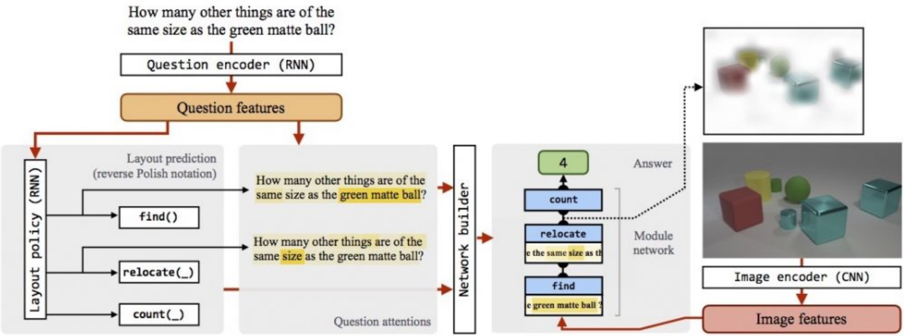


Figure 5: Learning to reason model overview [15].

Neural Module Networks (NMN): Evolution 2

- **Knowledge guidance** for intermediate modules.
  - **Bboxes** of relevant visual regions for attention modules.
  - KL divergence between predicted attention maps and knowledge guidance.

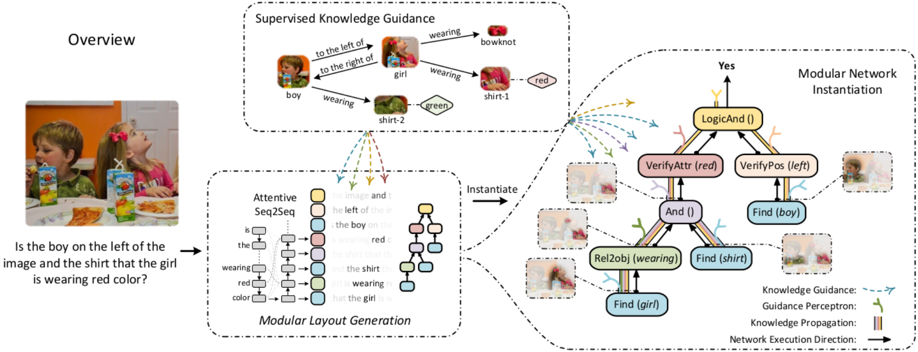
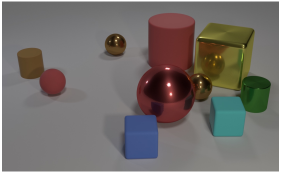


Figure 6: Perceptual Visual Reasoning overview Li, Wang, and Zhu [24].

Compositional Datasets: CLEVR

- Supervised learning task: Example = (Question, Program, Image, Answer).
  - CLEVR dataset [19]: synthetic images.



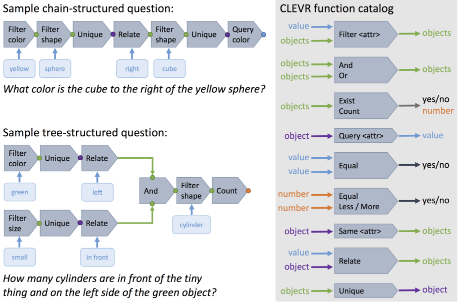


Figure 7: Image on the Left, Functional Program and Question on the Right.

Compositional Datasets: GQA

- Supervised learning task: Example = (Question, Program, Image, Answer)
  - GQA dataset [18]: real world images.

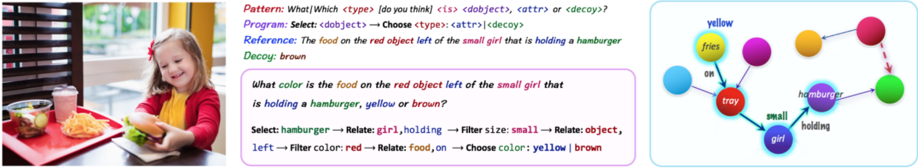


Figure 8: Left: Image, middle: Functional Program and Question, right: image graph.

Model	validation	testdev
BAN [22]	61.5	55.2
CTI [13]	61.7	54.9
MCAN [38]	-	57.4
MMN [11]	-	60.4
NMS [16]	-	63.2
HAN [21]	-	69.5
LXMERT [34]	59.8	60.0
OSCAR [26]	-	61.6
CFR [29]	73.6	72.1

Table 1: Accuracy results on the GQA dataset validation and testdev splits from [29].

Multi-modal Neural Module Network

- LXMERT ❄ [34] : extract **aligned cross-modal embeddings** for words and objects.
- Why VLP as features extractor ?
  - NMNs limitation in capturing language priors and commonsense knowledge [4, 5, 24].
  - VLPs are cross-modal foundation models for encoding text and images.
  - Focus on the downstream task of modular reasoning.
- Why LXMERT [34] ?
  - "Bottom-up features" have shown to significantly boost VQA performance [3].
  - Fine-tuned model on GQA is publicly available.

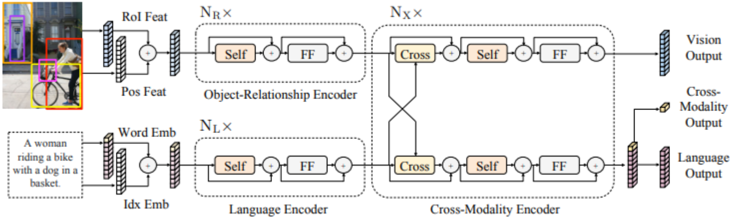


Figure 9: LXMERT: Learning Cross-Modality Encoder Representations from Transformers [34].

- **Generator**: Transformer decoder to decompose the question into a modules' program.
- **Executor**: instantiate and run the program over the image and answer the question.
- **Textual argument** to indicate the desired module's facet.
- **Dependencies**: the input of a module is its previous module's output.

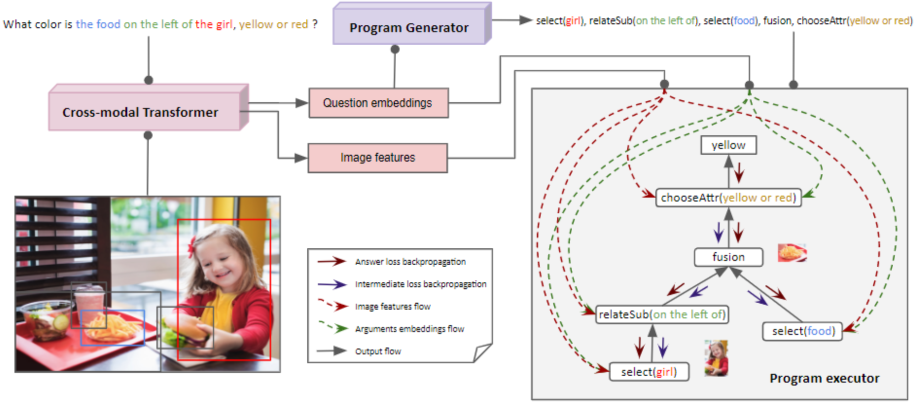


Figure 10: My framework [1, 2].

Modules taxonomy

- Programs preprocessing → To create a more manageable and concise catalog.
- From 136 functions to 29 modules.
- Consider functional and structural properties.
- Group rare modules into more general modules.
- Add new modules: `fusion`, `answerLogic`.

$$\left.\begin{array}{l}\texttt{filter color}\\ \texttt{filter size}\\ \texttt{filter material}\\ \texttt{filter width}\end{array}\right\}\texttt{filter attribute} \qquad \left.\begin{array}{l}\texttt{chooseHealthier}\\ \texttt{chooseOlder}\end{array}\right\}\texttt{compare}$$

- Modules are **shallow**, MLPs, products.
- **Weight sharing** to reduce the number of parameters.
- Three module groups: **attention, boolean, answer**.

Neural modules: attention

Name	Dependencies	Output	Definition
Select	–	attention	$x = r(W\,t), Y = r(WV)$ $o = \sigma(W(Y \odot x))$
RelateSub	[a]	attention	$x = r(W\,t), y = r(W(Va)), Z = r(WV)$ $o = \sigma(W(x \odot y \odot Z))$

Table 2: Sample module definitions. S: softmax, σ: [softmax,sigmoid], r: RELU, W*: weight matrix,* a*: attention vector (36×1),* V*: visual features (768×36),* t*: text features (768×1), $\odot$: Hadamard product.*

Question & Program prototype & Answer	Image
- Is the woman that is to the left of the bus wearing shorts? - `[select(bus), relateSub(to the left of), select(woman), fusion(), select(shorts), verifyRelObj(wearing), answerLogic()]` - yes	

Neural modules: boolean

Name	Dependencies	Output	Definition
VerifyAttr	[a]	boolean	$x = r(W\,t), y = r(W(V\,a))$ $o = \sigma(W(x \odot y))$
And	[b_1,b_2]	boolean	$o = b_1 \times b_2$

Table 3: Sample module definitions. S: softmax, σ: [softmax,sigmoid], r: RELU, W: weight matrix, a: attention vector (36×1), V: visual features (768×36), t: text features (768×1), $\odot$: Hadamard product.

<table>
<tr><th>Question & Program prototype & Answer</th><th>Image</th></tr>
<tr><td>- Are the grapes on top of the table and the fruits inside the box both red?

- <code>[select(table), relateSub(on top of), select(grapes), fusion(), verifyAttr(red), select(box), relateSub(inside), select(fruit), fusion(), verifyAttr(red), and(), answerLogic()]</code>

- yes</td><td></td></tr>
</table>

Neural modules: answer

Name	Dependencies	Output	Definition
ChooseAttr	[a]	answer	$\mathsf{x} = r(\mathsf{W}\,\mathsf{t}), \mathsf{y} = r(\mathsf{W}(\mathsf{V}\,\mathsf{a}))$ $\mathsf{o} = S(\mathsf{W}(\mathsf{x} \odot \mathsf{y}))$
QueryName	[a]	answer	$\mathsf{x} = r(\mathsf{W}(\mathsf{V}\,\mathsf{a})), \mathsf{o} = S(\mathsf{W}\,\mathsf{x})$

Table 4: Sample module definitions. S: softmax, σ: [softmax,sigmoid], r: RELU, W*: weight matrix,* a*: attention vector (36×1),* V*: visual features (768×36),* t*: text features (768×1), $\odot$: Hadamard product.*

Question & Program prototype & Answer	Image
- Is it outdoors or indoors? - `[select(scene), chooseAttr(outdoors indoors)]` - outdoors	
- Which kind of furniture is green? - `[select(furniture), filterAttr(green), queryName()]` - desk	

Generator

- Translation task: Questions in NL $\rightarrow$ functional programs.
- Transformer network in a decoder form [35].
- Program: $P = [m_1, ..., m_n]$.
- Modules prediction loss:

$$L_m = \frac{1}{T} \sum_{t=1}^{T} L_{CE}(\hat{m}_t, m_t^*) \tag{1}$$

- Arguments prediction loss:

$$L_a = \frac{1}{K} \sum_{k=1}^{K} L_{CE}(\hat{a}_k, a_k^*) \tag{2}$$

- Generator loss:

$$L_G = \alpha \cdot L_m + \beta \cdot L_a \tag{3}$$

- Instantiates the NMN based on the program.
- Executes it on the related image in a sequential manner to answer the question.
- Manages the module dependencies.
- Answer prediction:

$$\hat{y} = \arg\max_{y \in A} p(a|Q, P, I, \theta) \tag{4}$$

- Executor loss:

$$L_e = \frac{1}{N} \sum_{n=1}^{N} L_{CE}(\hat{y}_n, y_n^*) \tag{5}$$

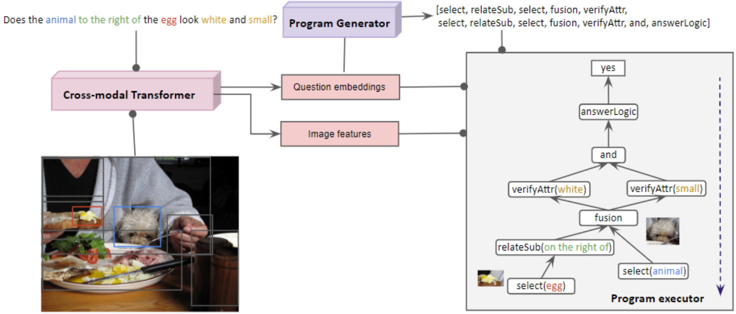


Figure 11: *My framework [1, 2]*

Unimodal vs Crossmodal representations

- **FastTextV**: Unimodal non-contextual fastText embeddings [10] and Faster-RCNN bboxes [18].
- **BertV**: Unimodal contextual Bert embeddings [12] and Faster-RCNN bboxes [18].
- **LXV**: Cross-modal representations encoded by the LXMERT model [34].

Embedding-training	Accuracy
FastTextV-setup1	0.495
BertV-setup1	0.506
LXV-setup1	**0.630**
FastTextV-setup2	0.511
BertV-setup2	0.485
LXV-setup2	**0.632**

Table 5: Language and vision representation comparison on `testdev-all`.

Guided Training of neural module networks

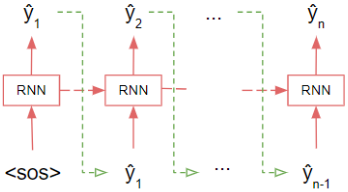


Figure 12: *RNN w/o TF*

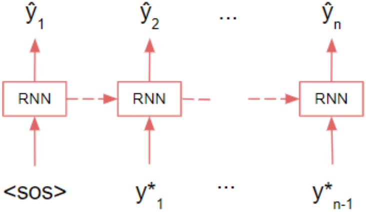


Figure 13: *RNN w/ TF*

- TF [36] is a widely used training technique in generation tasks.
- Instead of using the model's predicted output, use the true output from the previous step as input.

\+ Accelerates learning by providing accurate guidance.

– Exposure bias. Model isn't exposed to its own errors during training.

- Scheduled sampling (SS) [8]: At each step, randomly choose between using GT or model predictions.

- **Input guidance:** Decaying teacher forcing (TF).
  - Probability ϵ_e decreases as epoch number e increases.
  - Coin flip at each reasoning step (t).
  - Use predicted $\hat{o}_{t-1}$ as input with probability ϵ_e.
  - Use GT o^*_{t-1} as input with probability $1 - \epsilon_e$.
- **Output feedback:** multi-task (MT) loss $L = \alpha L_{att} + \beta L_{bool} + \gamma L_{answer}$

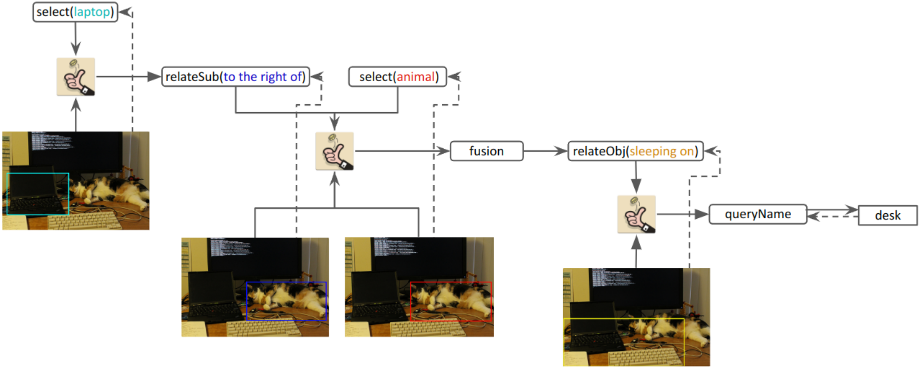


Figure 14: Q: 'On what is the animal to the right of the laptop sleeping?'.
*Teacher guidance: **input guidance** (Plain arrows) and **output feedback** (dotted arrows).*

Evaluation protocol

- Uni-modal vs cross-modal: **FasttextV**, **BertV**, **LXV**.
- **TF**: Decaying teacher forcing to guide the inputs of the modules.
- **MT**: Multi-task losses to guide the outputs of the modules.
- Matching techniques to align GT bboxes with detector bboxes:
  - **Hard** matching: Highest IoU, one target, CE (setup 1).
  - **Soft** matching: IoU $\geq$ threshold, multi-label, BCE (setup 2).

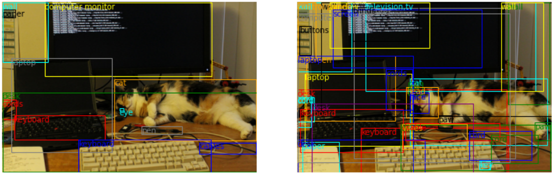


Figure 15: Left: GT bboxes, right: LXMERT bboxes.

Results analysis

Model	accuracy
LXV-TF-hard	0.548
LXV-MT-hard	0.598
LXV-TF-MT-hard	0.630
LXV-TF-soft	0.536
LXV-MT-soft	0.563
LXV-TF-MT-soft	**0.632**
FasttextV-TF-MT-hard	0.495
BertV-TF-MT-hard	0.506
BertV-TF-MT-soft	0.485
FasttextV-TF-MT-soft	0.511

Table 6: Performance of various training methods on the `testdev-all` set.

- LXV-TF vs. LXV-MT: MT achieves higher accuracy compared to decaying TF alone.
- Combination of TF and MT achieves highest accuracy: **LXV-TF-MT-soft** at 63.2%.
- Complementary effects: MT and decaying TF enhance training dynamics and performance.
- Cross-modal aligned features (**LXV**) yield accuracy improvements compared to unimodal features (**BertV**, **FasttextV**).

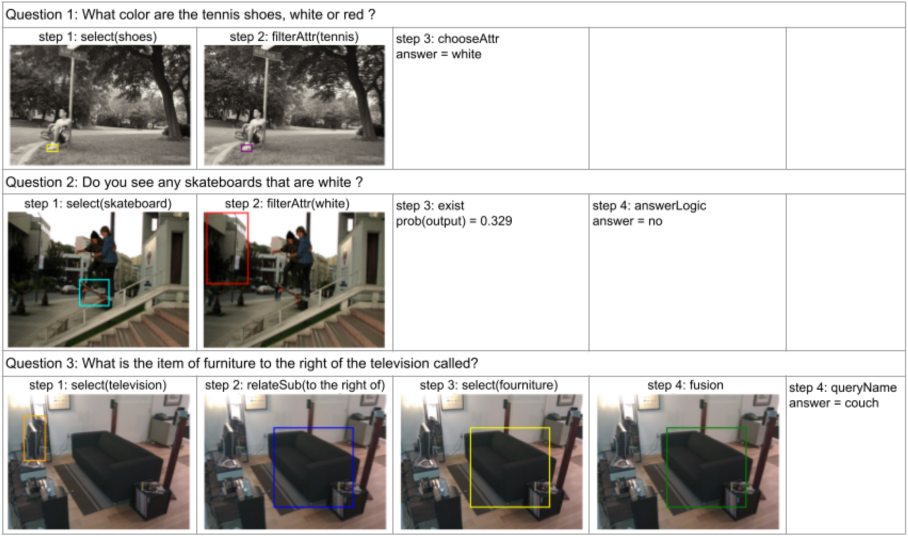


Figure 16: Examples showing the reasoning process.

Curriculum learning for neural module networks

Curriculum Learning CL

- CL is a **start small** training strategy similar to human learning [9].
- Start by easy training examples then gradually increase the examples' complexity.
- Applied to ML tasks and recently to text QA [27] and synthetic images VQA [7].
- Speed up convergence and use less training examples [9].

CL strategy definition

- difficulty criterion.
- Scheduler.
- Sampling function.
- Performance evaluator.

- **Difficulty**: Number of concepts in the question is the a priori for CL.
- CL difficulty refinement based on program length.

Question: Which color do you think the train car is?'

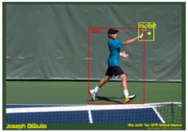
Program: select(<u>train car</u>), queryAttr(color).

Difficulty level: 1

Question: Is the racket to the right or to the left of the person in the middle of the picture?'

Program: select(<u>person</u>), filterPos(middle), select(<u>racket</u>), choosePos(to the left or to the right).

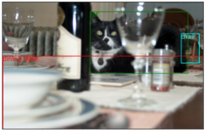
Difficulty level: 2

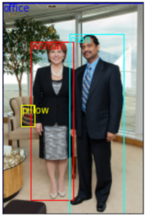


Question: What do you think is the piece of furniture to the right of the white animal that is lying on the dining table?'

Program: select(<u>dining table</u>), relateSub(lying on), select(<u>animal</u>), fusion, filterAttr(white), relateSub(to the right of), select(<u>furniture</u>), fusion, queryName.

Difficulty level: 3





Question: Do both the man in the office and the woman to the right of the pillow look happy?

Program: select(<u>office</u>), relateSub(in), select(<u>man</u>), fusion, verifyAttr(happy), select(<u>pillow</u>), relateSub(to the right of), select(<u>women</u>), fusion, verifyAttr(happy), and, answerLogic.

Difficulty level: 4

Figure 17: Dataset samples with different difficulty levels.

CL training protocol

- **CL scheduler**: update the curriculum every fixed sample size (1M examples).
- **sampling functions**:
  - Uniform sampling.
  - Balance the answer modules' occurences.
  - Program modules average loss.

* Sampling with replacement.

* Avoid catastrophic forgetting by retaining few previous examples (20%).

- GQA dataset: Balanced $\subset$ Unbalanced.

\- Train on the unbalanced split to have more examples.

Split	Train	Val	Testdev	Test	Challenge
Balanced	**943.000**	132.062	12.578	95.336	50.726
Unbalanced	**14.305.356**	2.011.853	172.174	1.340.048	713.449

Table 7: GQA dataset partitioning

Evaluated methods

- Without CL:
  - **Unbalanced**: Unbalanced GQA with random batch training (14M per epoch).
  - **Balanced**: Balanced GQA split with random batch training (1M per epoch).
  - **Random**: 1M random examples from unbalanced GQA at every iteration.
- **CL**: CL trainining strategy with increasing difficulty. 1M programs meticulously sampled from unbalanced GQA every iteration.
  - **Length (L)**: Filter by length (short, medium, or long) withing each difficulty level.
  - **Weights (W)**: Sampling: 'uniform', 'answer module'(W.a), 'modules loss'(W.b).
  - **Pretrain (P)**: Parameters initialisation from the 2nd iteration of Random variant.
  - **Repeat (R)**: Repeat the same CL-iteration twice.

Comparison of CL methods

Model	CL configuration			Iterations	Number of examples ($\leq$)	Accuracy
	weighting	pretraining	iterations/level			
CL+W.a	answer	–	1	4	4 M	0.642
CL+W.b	losses	–	1	4	4 M	0.635
CL+L	uniform	–	1	11	11 M	0.650
CL+L+W.a	answer	–	1	12	12 M	0.655
CL+W.a+P	answer	2 iterations	1	[2] + 3	5 M	0.670
CL+W.a+P+R	answer	2 iterations	2	[2] + 5	7 M	**0.681**

Table 8: Results on `testdev-all` for several CL strategies.

- 'answer' weighting **W.a** is the most effective weighting function.
- **CL+L** refinement improves the results over CL but the experiments are expensive.
- **P** "warms up" the model to the modular aspect of VQA framework.
- **R** helps to better learn the task without augmenting the data size.
- **CL+W.a+P+R** model is the **best modular VQA model** scoring 68.1% accuracy after 7 training iterations using less than 7M examples, *i.e.* less than half of the training data.

Impact of CL

Model	Comp. cost	# examples	Accuracy
Unbalanced	9×14 M	14 M	**0.702**
Balanced	50×1.4 M	1.4 M	0.678
Random	12×1 M	≤ 12 M	0.694
CL+W.a+P+R	7×1 M	< 7 M	0.681

Table 9: Comparaison of our CL model (CL+W.a+P+R) with no-CL models (Unbalanced, Balanced, and Random) on the `testdev-all` set. Computation cost is the number of seen examples per iteration times the number of iterations.

- Unbalanced model (14M) has the highest accuracy (70.2%) but has the highest cost.
- Balanced model achieves lower results than the Unbalanced mode.
- CL+W.a+P+R significant gains in terms of **computational cost** (18 times cheaper).
- CL+W.a+P+R needs half of the training # examples.
- CL+W.a+P+R the best trade-off between performance and training cost.

Conclusion & perspectives

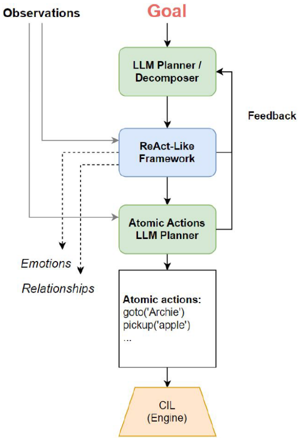
Conclusion

- Neural module network for visual reasoning in a **real world** VQA context [18].
- Decompose the reasoning task to a series of easier and more general sub-tasks.
- Transparency and interperability.

Contributions

- Benefit from **cross-modal representations** [34] for Compositional VQA.
- NMN trained with **Teacher guidance** to enhance model performance [2].
- **Curriculum Learning** to find a trade-off between cost and performance [1].

- CL for compositional action recognition in videos.
- Guided LLM prompting for multi-step reasoning in complex VQA tasks. Program = Python code.
  - ViperGPT [33], VisProg [14], and CodeVQA [32].
- Interesting applications for Video games using language-guided plan and execute strategies.
  - Actions = goto(object), pickup(object), attack(object), useon(object, object), talktoabout(object, msg)
  - Objects = (stone, hammer, John, tree, axe)
  - Goal = "Cut down the tree"
  - Plan:
    1. Goto(tree)
    2. Pickup(axe)
    3. Useon(axe, tree)
    4. Attack(tree)



Q & A

References

[1] Wafa Aissa, Marin Ferecatu, and Michel Crucianu. "Curriculum Learning for Compositional Visual Reasoning". In: *Proceedings of VISIGRAPP 2023, Volume 5: VISAPP*. 2023.

[2] Wafa Aissa, Marin Ferecatu, and Michel Crucianu. "Multimodal Representations for Teacher-Guided Compositional Visual Reasoning". In: *Advanced Concepts for Intelligent Vision Systems, 21st International Conference (ACIVS 2023)*. Ed. by Jacques Blanc-Talon et al. Springer International Publishing, 2023.

[3] Peter Anderson et al. "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering". In: *CVPR*. 2018.

[4] Jacob Andreas et al. "Learning to Compose Neural Networks for Question Answering". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 1545–1554. DOI: 10.18653/v1/N16-1181. URL: https://aclanthology.org/N16-1181.

[5] Jacob Andreas et al. "Neural Module Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 39–48.

[6] Stanislaw Antol et al. "Vqa: Visual question answering". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2425–2433.

[7] Narjes Askarian et al. "Curriculum learning effectively improves low data VQA". English. In: *ALTA 2021*. Association for Computational Linguistics (ACL), 2021, pp. 22–33.

[8] Samy Bengio et al. "Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks". In: *CoRR* (2015). arXiv: 1506.03099.

[9] Yoshua Bengio et al. "Curriculum Learning". In: ICML '09. Montreal, Quebec, Canada: Association for Computing Machinery, 2009, pp. 41–48. ISBN: 9781605585161.

[10] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *Transactions of ACL* 5 (July 2016).

[11] Wenhu Chen et al. "Meta Module Network for Compositional Visual Reasoning". In: Jan. 2021, pp. 655–664. DOI: 10.1109/WACV48630.2021.00070.

[12] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the NAACL: Human Language Technologies, Volume 1*. June 2019.

[13] Tuong Do et al. "Compact trilinear interaction for visual question answering". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 392–401.

[14] Tanmay Gupta and Aniruddha Kembhavi. "Visual programming: Compositional visual reasoning without training". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 14953–14962.

[15] Ronghang Hu et al. "Learning to Reason: End-to-End Module Networks for Visual Question Answering". In: *CoRR* abs/1704.05526 (2017). arXiv: 1704.05526.

[16] Drew Hudson and Christopher D Manning. "Learning by abstraction: The neural state machine". In: *Advances in Neural Information Processing Systems* 32 (2019).

[17] Drew A Hudson and Christopher D Manning. "Compositional Attention Networks for Machine Reasoning". In: 2018.

[18] Drew A. Hudson and Christopher D. Manning. "GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering". In: (2019).

[19] Justin Johnson et al. "CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning". In: *CVPR*. 2017.

[20] Justin Johnson et al. "Inferring and Executing Programs for Visual Reasoning". In: *ICCV*. 2017, pp. 3008–3017. DOI: 10.1109/ICCV.2017.325.

[21] Eun-Sol Kim et al. "Hypergraph attention networks for multimodal learning". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 14581–14590.

[22] Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. "Bilinear attention networks". In: *Advances in neural information processing systems* 31 (2018).

[23] Dan Klein and Christopher D Manning. "Accurate unlexicalized parsing". In: *Proceedings of the 41st annual meeting of the association for computational linguistics*. 2003, pp. 423–430.

[24] Guohao Li, Xin Wang, and Wenwu Zhu. "Perceptual Visual Reasoning with Knowledge Propagation". In: MM '19. Nice, France: Association for Computing Machinery, 2019, pp. 530–538. ISBN: 9781450368896.

[25] Liunian Harold Li et al. "VisualBERT: A Simple and Performant Baseline for Vision and Language". In: *Arxiv*. 2019.

[26] Xiujun Li et al. "Oscar: Object-semantics aligned pre-training for vision-language tasks". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX 16*. Springer. 2020, pp. 121–137.

[27] Cao Liu et al. "Curriculum Learning for Natural Answer Generation". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. IJCAI'18. Stockholm, Sweden, 2018.

[28] **Jiasen Lu et al.** "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks". In: *NeurIPS*. Ed. by Hanna M. Wallach et al. 2019, pp. 13–23.

[29] **Binh X Nguyen et al.** "Coarse-to-fine reasoning for visual question answering". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 4558–4566.

[30] **Ethan Perez et al.** "Film: Visual reasoning with a general conditioning layer". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[31] **Shaoqing Ren et al.** "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *NeurIPS*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028Chen%20et%20al.%20a21ed38046-Paper.pdf.

[32] **Sanjay Subramanian et al.** "Modular Visual Question Answering via Code Generation". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 747–761. DOI: 10.18653/v1/2023.acl-short.65. URL: https://aclanthology.org/2023.acl-short.65.

[33] **Dídac Surís, Sachit Menon, and Carl Vondrick.** "ViperGPT: Visual Inference via Python Execution for Reasoning". In: *arXiv preprint arXiv:2303.08128* (2023).

[34] **Hao Tan and Mohit Bansal.** "LXMERT: Learning Cross-Modality Encoder Representations from Transformers". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. 2019.

[35] **Ashish Vaswani et al.** "Attention is All you Need". In: *NeurIPS*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[36] **Ronald J. Williams and David Zipser.** "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks". In: *Neural Computation* 1.2 (June 1989), pp. 270–280.

[37] **Zichao Yang et al.** "Stacked attention networks for image question answering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 21–29.

[38] **Zhou Yu et al.** "Deep modular co-attention networks for visual question answering". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 6281–6290.

Executor

- Reduce the number of parameters.
- Weight sharing based on functional and architectural properties of modules:

Name	Definition
Select	$x = r(W\,t), Y = r(WV), o = \sigma(W(Y \odot x))$
FilterAttr	$x = r(W\,t), Y = r(WV), z = \sigma(W(Y \odot x)), o = \min(a, z)$
FilterNot	$x = r(W\,t), Y = r(WV), z = \sigma(W(Y \odot x)), o = \min(a, 1 - z)$
VerifyAttr	$x = r(W\,t), y = r(W(V\,a)), o = \sigma(W(x \odot y))$
RelateSub	$x = r(W\,t), y = r(W(V\,a)), Z = r(WV), o = \sigma(W(x \odot y \odot Z))$
RelateObj	$x = r(W\,t), y = r(W(V\,a)), Z = r(WV), o = \sigma(W(x \odot y \odot Z))$

Table 10: Weight sharing: shared layers have the same colors.
W*: weight matrix,* σ*: Activation function,* r*: RELU,* a*: attention vector (36 × 1),* V*: visual features (768 × 36),* t*: text features (768 × 1),* o*: attention vector (36 × 1)* $\odot$*: Hadamard product,* min*: element-wise minimum.*

Program examples

Question & Program prototype & Answer	Image
What is the fence made of? `[select(fence), queryAttr(material)]` wood	
Is it outdoors or indoors? `[select(scene), chooseAttr(outdoors indoors)]` outdoors	
Do these animals have different types? `[select(animal), differentAll(type), answerLogic()]` yes	

Question & Program prototype & Answer	Image
Is the woman that is to the left of the bus wearing shorts? `[select(bus), relateSub(to the left of), select(woman), fusion(), select(shorts), verifyRelObj(wearing), answerLogic()]` yes	
What is the name of the vehicle that is made of the same material as the lock? `[select(lock), relateAttr(same material), select(vehicle), fusion(), queryName(name)]` carriage	
Are there both cats and whales in this photo? `[select(cat), exist(), select(whale), exist(), and(), answerLogic()]` no	

LXMERT

- VLP model trained on various unimodal and multi-modal tasks.
- Image representations: region-based features extracted by Faster-RCNN object detector [31].
- BERT-like encoder blocks [12].
- Self-attention mechanisms for intra-modal relationships. $\text{SelfAtt}_{L \to L}$, $\text{SelfAtt}_{V \to V}$.
- Cross-attention mechanisms for inter-modal alignment. $\text{CrossAtt}_{L \to V}$, $\text{CrossAtt}_{V \to L}$.

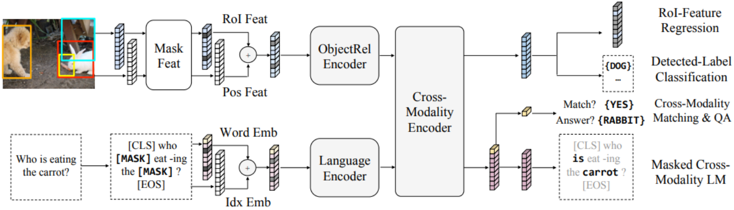


Figure 18: LXMERT [34].

Recap

- NMN architecture: module taxonomy, generator and executor.
- Modular reasoning $\rightarrow$ transparency.
- VLP features to capture information from language and vision and align them.
- How to improve executor training ?

Recap

- NMN trained with **Teacher guidance** to enhance model performance.
- input guidance: prevent error propagation during the early stages of training.
- Output feedback: optimize modules to perform their specific sub-tasks.
- Modules learn their reasoning sub-tasks both independently and collaborativly.

Recap

- NMN trained with CL.
- **Number of concepts** is an adequate CL difficulty criterion.
- By applying the appropriate CL we reduce experimental cost.
- Find a **trade-off between cost and performance.**