V2A - Vision to Action: Learning robotic arm actions based on vision and language

Michal Nazarczuk[1][0000−0003−1692−4470] and Krystian Mikolajczyk[1][0000−0003−0726−9187]

Imperial College London, London, United Kingdom
{michal.nazarczuk17,k.mikolajczyk}@imperial.ac.uk

Abstract. In this work, we present a new AI task - Vision to Action (V2A) - where an agent (robotic arm) is asked to perform a high-level task with objects (*e.g.* stacking) present in a scene. The agent has to suggest a plan consisting of primitive actions (*e.g.* simple movement, grasping) in order to successfully complete the given task. Queries are formulated in a way that forces the agent to perform visual reasoning over the presented scene before inferring the actions. We propose a novel approach based on multimodal attention for this task and demonstrate its performance on our new V2A dataset. We propose a method for building the V2A dataset by generating task instructions for each scene and designing an engine capable of assessing whether the sequence of primitives leads to a successful task completion.

1 Introduction

Our goal is to develop an intelligent agent capable of perceiving and reasoning about the environment by combining information from different modalities. Visual Question Answering (VQA) [1] is a related task that requires combining language and visual input. Recently, Embodied Question Answering (EQA) [2] was proposed as a new challenging problem incorporating navigation as the necessary step for successful question answering. We propose a new task, Vision to Action (V2A), to bridge the gap between visual reasoning and action planning which requires the agent to perform a multimodal perception. We introduce a dataset generated for this purpose, along with an engine capable of assessing the correctness of predicted sequences of actions.

VQA and EQA tasks involve an agent to provide a specific answer to a given question. For the question: *What is the weight of the big blue box?*, the answer can be *heavy*. Humans typically interact with the object before answering such question. We gain knowledge about the properties of the objects in the world through exploration as well as from the external knowledge. When asked a given question, our answer may be *let me check that in the Internet* or *let me hold it to estimate the weight, or use a scale* for a more accurate answer. In our work, we focus specifically on the latter, planning the exploration using simple, primitive actions. We consider a scenario with objects on a table in the context of robotic manipulation. Programming a manipulator is a similar task to composing

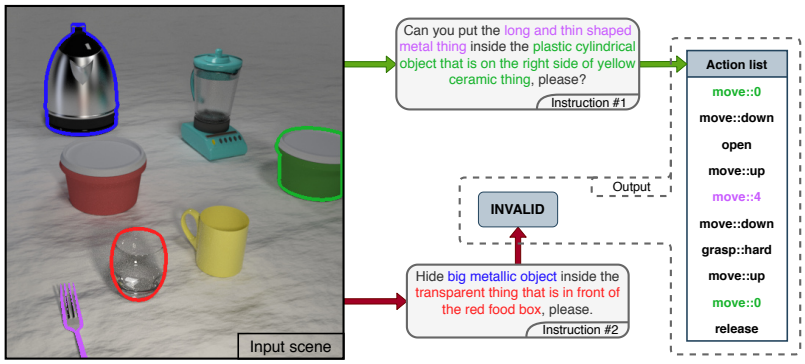


Fig. 1: Vision to Action task. Given the scene and instruction, the task is to predict a sequence of primitive actions leading to a successful completion of the task. In order to enforce multimodal perception, data includes instructions correct linguistically, although some not possible to complete given the scene.

a sequence of primitive actions. In addition, we claim that the tasks of visual reasoning and planning should be done jointly - a plan cannot be made based on the instruction only, but must rely on the current state of the environment.

The main focus of EQA is to include navigation in the task of Visual Question Answering. While its goal is to perform on-line prediction of the best exploration path, our proposed V2A is concerned with manipulation actions. Our main objective is to maximise information gain from the visual and language input such that, in combination with experience gained through exploration, the system is capable of assessing the validity of the given query and preparing a plan which can then be implemented in a real-world or simulated scenario.

V2A considers semantically high-level primitives and high-level description of the scene, which decouple the model from a particular robotics hardware or a simulated environment. Nonetheless, we keep the primitive actions similar to a high-level programming language, with the goal of being able to transfer the sequences into the real settings by wrapping primitives in code functions specific to the equipment. Realistic scenarios often require action planning that cannot be replaced with an online decision making process *e.g.* picking up lighter out of two objects requires planning to measure the weight of both of them. In such a situation, one needs to plan a sequence of action based on the current observation. An overview of the V2A task is presented in Figure 1. Given the input scene and a textual instruction, the system predicts the sequence of the actions, if the task can be performed given the scene. Otherwise, the system indicates invalidity of the task. To this end, we provide a new dataset for the V2A task along with an engine to assess the completion of a given instruction. Our dataset is inspired by the common robotic setup including a manipulator placed on a tabletop equipped with a camera. We also propose an evaluation protocol consisting of different metrics to assess different aspects of model performance. Finally, we propose a novel approach to perform the V2A task. Our approach incorporates a

Multimodal Attention Model that is used to simultaneously translate sequences of attribute vectors (corresponding to objects in the scene) and natural language instructions into a sequence of primitive actions that fulfils the given task.

2 Related work

Our work involves visual and linguistic reasoning, and action planning , intended for robotics applications. It is a new problem and there are no existing methods that are suitable for this task, however the relevant literature include papers from the area of VQA, EQA and action planning that we review below.

Visual Question Answering [1] is the task of providing the precise answer to a given question that is related to a given image. VQA follows two main approaches: projecting images and text onto a common subspace to predict the answer [3–5], or disentangling the scene and question into understandable forms to perform a human-like reasoning to discover the answer. The latter emerged as a new task, called Visual Reasoning, which along with predicting the correct answer, attempts to produce a logical reasoning that resembles a human approach to the problem. Neural Module Networks [6–8] propose to stack network modules to predict the flow of the focus over the image objects, and therefore explain the origin of the final answer. NS-VQA [9] and NS-CL [10] focus on fully disentangling the scene into a human-readable form, such that the reasoning can be performed by executing symbolic programs on the new representation.

Embodied Question Answering [2] adds an additional component to the VQA - navigation in a simulated environment. An agent is asked a simple question and its task is to navigate to the correct location and answer the question after. Current methods [11] split the task into two streams: navigation and question answering. The agent switches between modules to gather clues necessary to answer the query. The task of navigating an agent based on language and visual input was further explored for both simulated and real world scenarios [12–14].

Action planning focuses on producing a plan consisting of simple components based on the given input. A common source of the information about how to perform the action is a linguistic description. Detailed textual instructions on how to perform a task are translated into a series of actions in [15]. Generating plans for a high-level task in the setting of incomplete world knowledge is proposed in [16]. Video action recognition is used in [17] to generate sequences of actions corresponding to the presented task.

Available datasets include examples with multimodal data, but none of them have annotation in form of instruction/image pairs accompanied with ground truth sequences of actions. Furthermore, there are no tools to assess the correctness of predicted actions. The most relevant datasets are Visual Reasoning benchmarks. CLEVR [18] provides a number of simple synthetically generated

scenes composed of spheres, cylinders and cubes. CLEVR includes questions, answers and programs to obtain the answer given the scene graph. SHOP-VRB [19] builds upon CLEVR and provides a more realistic scenario of a tabletop with common household items. In addition to CLEVR structure, it provides some simple text descriptions for the objects along with questions grounded in the image and text simultaneously. SHOP-VRB also addresses the problem of reasoning about unseen instances of object classes. On the other end of the spectrum there exist robotics datasets concerning the performance of different tasks in simulated environments. DeepMind Control Suite [20] provides a number of control tasks for different types of robotic agents. ALFRED [21] focuses on language in robotics, as it provides detailed task descriptions along with action sequences grounded in that text that are to be executed in a simulated environment. RoboTurk [22] is mainly focused on imitation learning, as it contains a large base of different tasks paired with crowdsourced demonstrations for each. Surreal [23] provides a reinforcement learning framework for manipulation tasks.

V2A attempts to bridge the gap between the visual reasoning task, robotics action planning and available datasets. It is based on SHOP-VRB annotations, allowing for easy generation of new samples. It provides an engine that works on scene graphs that allows for very quick executions. The queries force the model to perform multimodal perception. Unlike ALFRED [21], that provides detailed descriptions linking high level tasks to sequence primitives, we allow the model to learn visual-action dependencies through exploration. Additionally, V2A remains very challenging from visual reasoning standpoint.

3 Dataset and engine

V2A involves generating a sequence of actions that, performed on a given scene, would lead to a successful completion of a given instruction or question. Hence, we introduce a new dataset containing natural language instructions along with an engine that allows for testing solutions and validating the result of the performed actions. We build on closely related SHOP-VRB datasets [19] that has recently been introduced for VQA in robotics.

3.1 Engine

The engine operates on a level of abstraction, that allows to decouple the task from real machines or specific simulating environments. It works as a state machine operating on two main abstractions - state of the scene, and state of the robotic arm. The dictionary of the state extends the one in SHOP-VRB scene graphs state by adding properties related to object manipulations (*e.g. is opened*, *current location*). The manipulator dictionary describes the current state of the manipulator and the gripper (*e.g. gripper closed*, *current location*).

Our engine performs all operations by calling a sequence of primitive actions on the scene and manipulator states. Primitive actions, presented in Table 1,

Table 1: Primitive actions in the proposed engine. The primitives are composed of action verbs, and parameters if needed. Parameters can include an index, a direction of movement or a property characteristic to the given action. If not specified, the action is performed on the object in focus.

Command	Argument	Functioning
avoid	index	Avoid object *index* during next move.
move	index	Move arm towards object *index*.
move	direction	Move arm in cardinal *direction*.
move	[up/down]	Lift or lower the arm.
grasp	[soft/hard/attach]	Grasp the object next to the arm using given grip.
release	-	Release object from the gripper.
open	-	Open the object next to the arm.
close	-	Close the object next to the arm.
measure	property	Measure *property* of the object next to the arm.
shake	-	Shake the object in the gripper.
rotate	-	Rotate the object in the gripper.
flip	-	Flip the object in the gripper.
button press	-	Press the button on the object next to the arm.
fill	-	Fill the object next to the arm.

require the scene and the manipulator to be in particular states to be possible to execute, and modify these states according to the performed action. Construction of primitives and operation on state dictionaries allow new tasks to be easily added to the engine, simply by specifying the required conditions and changes in the state due to the action performed. As proposed in Table 1, primitive actions take the form of operations related to robotic manipulation. The primitives are on high semantic level in order for the benchmark to avoid limitations from any specific language. It is assumed that any presented action can be wrapped in a software-specific function such that, given the proper parameters, it would result in similar changes in the scene as we describe. The parameters of the scene and the manipulator that are required for the primitive action, reflect the behaviour of a robotic arm. The set of requirements and changes are effectively creating the rules of the environment for the agent. For example, when moving the manipulator towards an object, one has to plan the trajectory such that the arm does not collide with any other object (*avoid::[close neighbors]* has to precede *move::down*). Similarly, when pushing the object by sliding it on the tabletop, the trajectory should avoid the neighbors in the movement direction. On the other hand, when the object is lifted, there is no need to avoid obstacles. Furthermore, when putting an object into a container with a lid, one has to open it to put the object in. Similar rules apply to all primitive actions presented.

The proposed engine is equipped with a functionality of checking whether the agent, given a task, can succeeded in completing it by preforming the sequence of inferred primitive actions. All benchmarking tasks are presented in Table 2.

Table 2: List of benchmark tasks available in V2A. The tasks, together with visual reasoning cues, are the base for the provided instructions. Each task is characterised with a different engine state that allows to assess whether the task can be completed with a given sequence of primitive actions.

#	Task	#	Task
1	Pick the *OBJ* up	12	Check if there are moving parts inside the *OBJ*
2	Open the *OBJ*	13	Check what is on the other side of the *OBJ*
3	Measure weight of the *OBJ*	14	Pick the *OBJ* up and move it *DIR*
4	Measure stiffness of the *OBJ*	15	Pick the *OBJ1* up and move it over the *OBJ2*
5	Measure roughness of the *OBJ*	16	Push the *OBJ DIR*
6	Measure elasticity of the *OBJ*	17	Push the *OBJ1* towards the *OBJ2*
7	Empty the *OBJ*	18	Stack the *OBJ1* on top of the *OBJ2*
8	Fill the *OBJ* with liquid	19	Put the *OBJ1* in the *OBJ2*
9	Turn the *OBJ* on	20	Hide the *OBJ1* inside the *OBJ2*
10	Boil the liquid inside the *OBJ*	21	Flip the *OBJ1* and shake it over the *OBJ2*
11	Shatter the *OBJ* (drop it)	22	Rotate the *OBJ1* and put it in the *OBJ2*

Due to the construction of the engine, checking the success is simply comparing the state of the manipulator and the state of the scene to the model state. Such an approach also makes the addition of more tasks very simple - by setting requirements on the final states of the scene and the manipulator. For example, when picking up an object, the target action is to simply change the state of the object to *picked up*. When hiding an object in another object, a container, one has to verify whether both are at the correct positions, and the lid of the container has been closed after putting the object in.

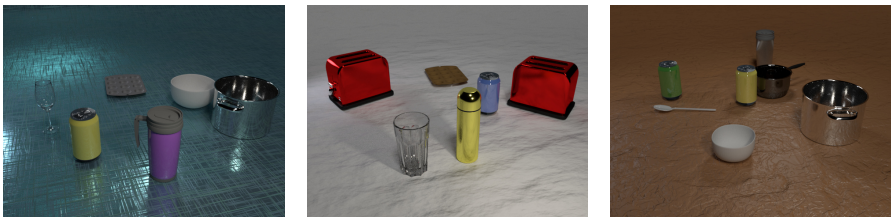
The ground truth action sequences for each given task are generated by the engine based on *prototype* sequences provided by human annotators. Prototypes contain a series of primitives always necessary to be performed to complete the given task (*e.g.* picking the object up has to contain *moving towards* and *grasping*). Based on the prototype along with the knowledge of the environment (required states of the primitives) the instructions are provided additional primitives such that the task is ultimately executed with success. We consider this ground truth to be the shortest sequence to achieve the goal for each task.

Due to the possibility of multiple correct solutions the ground truth sequences are intended for measuring only the efficiency of the system, and not for a supervised training of a predictor. Alternatively, such ground truth can be used for initialising a model that is then trained on other data.

3.2 Dataset

We generate a V2A dataset of challenging instructions grounded in visual inputs. We use the same scenes as in [19]. For each scene we provide 9 to 10 visually grounded instructions asking the agent to perform one of the tasks specified in

Table 2. Each instruction is equipped with a corresponding task, as well as the objects and parameters concerned by the task. Figure 2 contains sample scenes and corresponding instructions from V2A dataset.



(a) Could you push the medium-sized light metal portable thing behind the light yellow metal portable object on the right side of the table, please?

(b) Pick up and move the metal portable cylindrical thing that is on the left side of the medium-sized blue metal cylindrical thing over the brown portable thing.

(c) There is a medium-sized light object that is both on the right side of the white bowl and left of the portable thermos; could you hide it inside the heavy metallic object, please?

Fig. 2: Example scenes and queries from V2A dataset. Note the complexity of visual reasoning process required to successfully plan the set of primitive actions that fulfils the instruction.

Dataset instructions are generated in a procedural way. We employ 77 different instruction templates. Each template corresponds to one of the aforementioned tasks. Templates are chosen randomly for generating instructions while keeping the distribution of instructions over templates uniform. Each template is used to instantiate valid instructions, such that there exist a sequence of actions resulting in a successfully executed task, as well as invalid instructions, such that no sequence is able to lead to a positive result for a given task (*e.g.* asking to turn on a fork). Invalid instructions require the system to specifically indicate that a query is invalid (not only producing any invalid primitives sequence). Distribution of valid and invalid instructions is also uniform over the templates. While being generated, each new instruction is tested for correctness, and both the task and the concerned objects are returned as the answer. This allows the engine to test any generated sequence for execution correctness given the task, as well as to generate the ground truth sequence of primitives. Furthermore, the instructions provide a challenge for visual perception as they describe objects by their properties and their relations to other objects. Finally, functional programs, similar to those in CLEVR are also provided as a part of the ground truth.

V2A includes 4 splits: *training*, *validation*, *test* and *benchmark*. The *benchmark* split is comprised of scenes that contain the same classes as, but different instances of the objects as the other splits. In Table 3, we provide details of the V2A dataset containing the number of samples in each split.

Table 3: Number of instructions in V2A dataset for valid and invalid tasks. Note that some scenes do not contain enough objects to generate 10 distinct questions, hence, some of them contain fewer.

Split	training	validation	test	benchmark
valid	49965	7500	7500	7500
invalid	49911	7487	7494	7492
total	99876	14987	14994	14992

4 Multimodal Attention Model for V2A

We propose a three step approach as a Multimodal Attention Model for the V2A task. Additionally, we analyse the performance of a simple translation model in a blind scenario (without visual input).

Our proposed approach is inspired by NS-VQA [9]. It consists of three main components - scene segmentation, attributes extraction and multi-modal sequence-to-sequence instruction to action translation. The approach is illustrated in Figure 3.

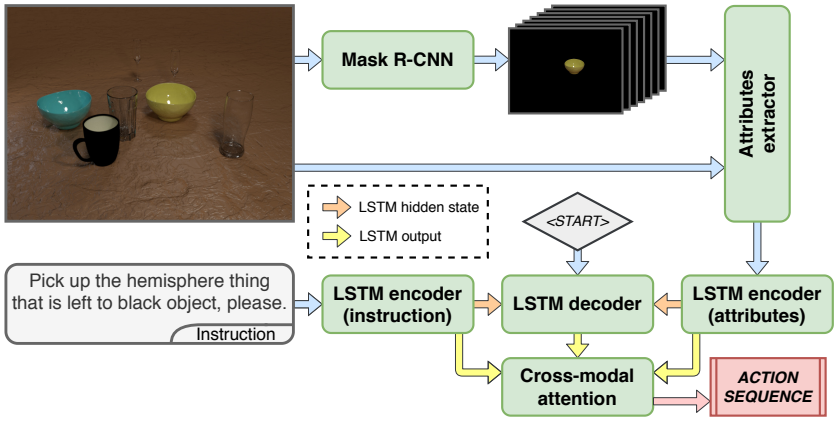


Fig. 3: Proposed Multimodal Attention Model. Input image is segmented with Mask R-CNN and processed by attributes extraction network to provide a disentangled representation of the scene. Both, attributes and a text instruction are parsed by LSTM encoders in order to decode the final instruction sequence using an LSTM decoder with a cross-modal attention layer.

Scene segmentation is performed using Mask R-CNN [24]. The network predicts the segmentation mask and category of each object.

Attributes extraction is implemented with the use of ResNet-34 [25]. Masked images of objects obtained from the segmentation step are concatenated with the full image and passed through the network to predict their properties and coordinates on the supporting plane. Scene segmentation and attributes extraction are trained with the ground truth provided with the scenes, resulting in a disentangled representation of the visual scene.

Instruction to action translation is based on the Seq2seq machine translation scheme [26]. Disentangled attributes vectors of the scene corresponding to the input instruction are passed as a sequence to the Bidirectional LSTM [27] encoder. Considering the input as a sequence of objects allows the model to point towards particular items within the scene. Similarly, instruction words are embedded to vector representations and passed through a second Bidirectional LSTM network. Further, hidden states for both representations are concatenated and set as the hidden state of the decoder network, which is a unidirectional LSTM. Both outputs from the image and the instruction LSTMs are projected with linear layers to match the output of the decoder LSTM. An attention mechanism is then applied to the pair of outputs from the image encoder-decoder output and the instruction encoder-decoder. The attention is implemented in a similar manner to that proposed in [28] and adopted in NS-VQA. Given the output decoder vector $\mathbf{q}_t$ for timestep t, and the encoder output $\mathbf{e}_i$ for timesteps i a new context vector for timestep t, $\mathbf{c}_t$ is calculated by a weighted sum of the encoded states according to the attention distribution:

$$\mathbf{c}_t = \sum_i \alpha_{ti}\mathbf{e}_i \qquad \text{where:} \qquad \alpha_{ti} = \frac{\exp(\mathbf{q}_t^\mathsf{T}\mathbf{e}_i)}{\sum_k \exp(\mathbf{q}_t^\mathsf{T}\mathbf{e}_k)} \tag{1}$$

The context vector is concatenated with the decoder output and projected back to its original size inside the attention layer. New representations for the attributes-focused decoding and the instruction-focused decoding are projected such that, after concatenation, their dimensions match those of the original decoder output. Finally, the obtained multimodal vector is projected onto the output vocabulary space in order to predict the next action token for the final action list. A detailed view of the decoder structure along with the attention mechanism is presented in Figure 4.

The training regime for translating instructions to actions is divided into two steps: weakly-supervised and reinforced. The weakly-supervised part of the training uses randomly selected ground truth action lists for the given instructions. Selected actions are uniformly distributed among the instruction templates, while keeping the ratio of valid to invalid templates at a given value. A total number of ground truth annotated instructions for the supervised part of the training does not exceed 3% of the whole training set size.

The reinforced step follows the weakly supervised part. We use REINFORCE [29] to further train the model on the full set of instructions. The reward is generated based on the execution of the generated action list in V2A engine. We allow the setting of different rewards for valid and invalid instructions.

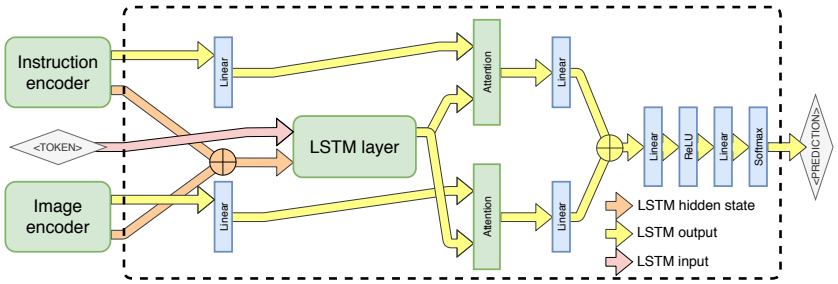


Fig. 4: Detailed view of the decoder and attention layers. Hidden states of both encoders are concatenated and used to initialise the decoder state. Outputs of the encoders are projected to the size of the decoder output in order to use the attention mechanism that produces context-aware outputs. These are further concatenated and used to predict the next token in action sequence.

5 Evaluation protocol

Success rate (SR) is a natural way of evaluating any agent performance in interactive tasks. It is defined as the ratio of successful attempts to all attempts. In our case, it is the ratio of correctly evaluated sequences to all given queries. However, we argue that such simple information is not sufficient to assess the performance of the agent in detail.

Valid success rate (VSR) is therefore a new metric we propose. By considering queries for which there exist a valid answer, we measure how well the model is able to generate successful plans.

Invalid success rate (ISR) is the complement of VSR and allows assessment of to whether the system is not overfitting to seemingly easier queries which do not have a valid answer. Considering the simple scenario of creating model that outputs only invalid tokens, one would obtain SR close to 0.5 (with the V2A valid/invalid ratio of roughly 1:1) but score exactly 0.0 in VSR.

Harmonic success rate (HSR) is our proposal of a metric combining the aforementioned scores into one based on the harmonic mean of VSR and ISR (being an analogue of F_1 score for precision and recall):

$$HSR = 2\,\frac{VSR \cdot ISR}{VSR + ISR} \tag{2}$$

Efficiency of the sequence (E_{score}) prevents the agent from *gaming* the engine by outputting all the possible moves that do not break the sequence (*e.g.* avoiding all obstacles one by one before every move). Efficiency is calculated as the ratio of the length of the primitives list generated by the model to the ground truth one (the shortest possible). Efficiency is calculated only for valid entries for which the model performed a successful task completion.

Action sequence accuracy (A_{acc}) assesses the accuracy of the focus on the given task by the model. To obtain the score, we calculate the ratio of successful task completions to all actions produced by the model that lead to correct execution according to the engine (with no errors, calculated only for valid instructions).

All suggested metrics, with information whether the goal is to maximise or minimise each score, and bounding values for each are presented in Table 4.

Table 4: Summary of the evaluation metrics with bounding values and indicated maximisation or minimisation goal.

Metric	SR	VSR	ISR	HSR	E_{score}	A_{acc}
better	↑	↑	↑	↑	↓	↑
min	0.0	0.0	0.0	0.0	1.0	0.0
max	1.0	1.0	1.0	1.0	∞	1.0

6 Experiments

We perform experiments on the *test* and *benchmark* splits of V2A dataset. *Test* split consists of new scenes generated with the same instances of objects as *train* and *validation* splits, whereas *benchmark* split challenges the system with scenes generated with the same classes of objects but new instances. We consider *benchmark* split to be an analogue of zero-shot learning capable of evaluating generalisation properties of the system in an unknown setting. We provide results for our two models, both blind and multimodal. Both models follow the training regime of supervised pretraining and reinforced fine-tuning. The amount of data sampled for the supervised part is the same for both models and does not exceed 3% of all *training* split instructions.

6.1 Blind model

Additionally to our full model, we propose an ablation evaluation with only the language part of the model. The method becomes a standard sequence-to-sequence model trained with the same regime as the full model (weakly-supervised + reinforced). The ablation study is proposed to show the importance of correct visual perception in the V2A task.

Seq2seq blind model is trained only with instruction-action sequence pairs for the supervised part and with instructions and engine rewards for the reinforced part. The supervised part uses negative log-likelihood as the loss function and is trained with Adam [30] with learning rate $1e-3$ as an optimiser. We use *validation* split as an early stopping criterion. For the reinforced part, learning rate is decreased to $1e-5$ and the reward is set to be 0.2 for invalid queries and 5.0 for valid ones.

6.2 Multimodal attention model

Multimodal attention model uses an attribute extraction pipeline similar to that in NS-VQA [9] . Firstly, we train Mask R-CNN for 30000 iterations with the supervision of ground truth masks. Thereafter, we train the attributes extractor, a ResNet-34, using *training* split. We apply early stopping based on *validation* split. In order to draw conclusions on the action planning for *test* and *benchmark* splits, we provide the attributes extraction accuracy for V2A scenes in Table 5. We train our multimodal action planner similarly to the blind model. We use the

Table 5: Attributes recognition accuracy for V2A scenes.

Split	Category	Size	Weight	Colour	Material	Mobility	Shape	Overall	Dist err
Test	88.3	88.9	88.9	88.5	88.4	89.2	88.7	88.7	0.062
Bench	43.2	61.2	53.7	50.8	48.8	65.4	38.0	51.4	0.102

same optimisers and loss functions, and keep rewards the same. Additionally, we train the model with ground truth attribute vectors as well as those inferred from the attributes extraction network in order to assess how the quality of features affects performance.

6.3 Results

The results for the aforementioned models are presented in Table 6.

Table 6: Results for V2A task. Text LSTM is a blind model without visual input, MAM refers to the full proposed approach, *test* and *benchmark* refer to the splits of the dataset and GT and *inferred* correspond to use of attributes from ground truth or extracted via the attributes extraction network, respectively.

Model	SR	VSR	ISR	HSR	E_{score}	A_{acc}
Text LSTM test	32.1	3.1	61.1	5.9	1.09	12.4
Text LSTM benchmark	31.9	2.5	61.4	4.8	1.08	12.6
MAM test GT	44.9	24.9	65.0	36.0	1.04	38.4
MAM test inferred	44.7	23.8	65.0	34.8	1.04	36.3
MAM benchmark GT	42.3	10.7	73.9	18.7	1.02	16.2
MAM benchmark inferred	40.2	6.4	79.9	11.9	1.02	10.2

The blind model performs very poorly overall which is the expected behaviour. One can quickly notice that with no visual input the challenge of *benchmark* split is not presented to the model, hence, the results are very similar for both *test* and *benchmark*. As explained in the evaluation protocol, one can expect balancing valid and invalid recognition to be a complex task. It can be inferred that the blind model has overfitted to be predicting a majority of sequences with invalid tokens as the answer. However, by looking at VSR and HSR scores one may instantly notice that in fact, the blind model was not able to learn the actual planning. Based on E_{score}, we can notice that correctly predicted action plans were not much longer that ground truth ones. A_{acc} is showing that the model was able to produce more programs that were executed by the engine with no errors (which means they can be performed on the given scene), however, only a small fraction of them were focused on the given task.

MAM system preforms significantly better than the blind one. We observed that both models were performing similarly for invalid queries. However, the multimodal attention model generated many more correct sequences for instructions with a valid answer. Increase in VSR drastically improved the harmonic mean HSR, proving it to be a very useful metric to assess the performance of various systems in the V2A task. In case of the multimodal approach, we observe a significant difference between *test* and *benchmark* splits. Seemingly, presenting a new, never-before-seen combination of attributes poses a nontrivial challenge to the system. One can notice that ISR improved for *benchmark* split with respect to *test* split. It may be inferred that the model learned to associate specific attributes as preventing the instruction to be fulfilled and such ones were associated with the valid instructions in *benchmark* split. Furthermore, all models seem not to add too many additional instructions to the ground truth action sequence (E_{score}). This is possibly due to the supervised pretraining step, presenting the system with some shortest model sequences. We can see a clear difference in action sequence accuracy A_{acc} between *test* and *benchmark* splits. This indicates that the model was more precise in performing the task correctly for *test* split, which may be caused by misleading attributes from *benchmark* split. Finally, one may notice a significant impact of the attributes recognition accuracy on the given model. A difference in the performance between models using the ground truth attributes and the attributes extracted by the network correspond to attributes recognition accuracy (Table 5. We observe bigger gap in performance for *benchmark* split, which attributes were extracted with much lower accuracy than for *test* split.

Breakdown of success rates among different task with distinction for valid and invalid queries was also investigated. In Figure 5 we present the breakdown of success rates, keeping the numbering consistent with Table 2. We observe that the distribution of accuracies is not even among the tasks, especially for valid samples. One can notice a significant decrease in performance for tasks 14 to 22. Those tasks correspond to instructions asking to manipulate more than one object within the same sequence. It is then not surprising that the system finds

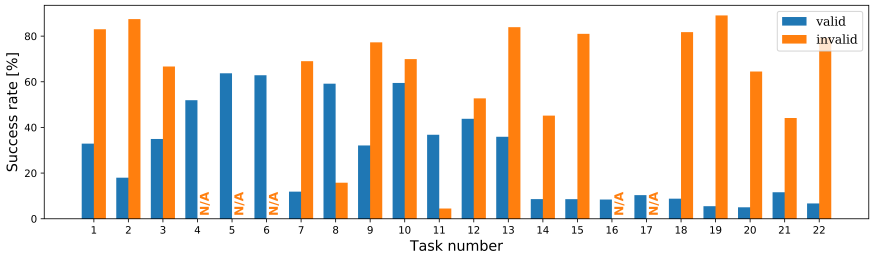


Fig. 5: Breakdown of success rates for the full model (experiment on *test* split). Task numbers refer to Table 2. Note that some tasks can always be performed no matter the scene given (*e.g.* we assume objects cannot always be picked up but can always be pushed), hence some columns are not present for *invalid* part.

such a task harder than manipulating a single object, as it also follows the human intuition of being a harder task.

7 Conclusions

In this work, we presented a new challenging task for experiments in computer vision and high-level action planning. We believe that V2A bridges the gap between visual and language reasoning and action planning, and in particular, object manipulation with a robotic arm. We show that the task poses a significant challenge for multimodal perception. An agent has to be developed with the understanding of natural language instructions that are grounded in an image - visual input. The grounding forces the planning to be performed in a multimodal space. We believe that V2A defines a task that resembles modalities that are offered to an agent working in a real-world environment.

Along with the task, we provide a dataset containing natural language instructions for an agent, and an engine capable of assessing whether the task has been performed successfully. The engine operates on functional primitives that are actions assigned to be performed by the robotic arm. Primitives operate on a high level of abstraction as we want to decouple the task from any particular hardware or simulation software. Additionally, we believe that such an approach enables further integration with a real system by wrapping primitive actions in a syntax specific to a terminal system.

Our experiments proved V2A to be a challenging, but well-defined task. We observe that challenges that are naturally hard for a human are also hard for the system (manipulating more objects in a sequence is much harder than manipulating one). An experiment with a blind system trying to learn primitive action sequences directly from instructions shows that visual perception is an integral part of the task, and must necessarily be addressed when trying to solve it.

Acknowledgements. This research was supported by UK EPSRC IPALM project EP/S032398/1.

References

1. Agrawal, A., Lu, J., Antol, S., Mitchell, M., Zitnick, C.L., Batra, D., Parikh, D.: VQA: Visual Question Answering. In: International Conference on Computer Vision (ICCV). (2015)
2. Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., Batra, D.: Embodied Question Answering. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2018)
3. Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., Zhang, L.: Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2018)
4. Kim, J.H., Jun, J., Zhang, B.T.: Bilinear Attention Networks. In: Advances in Neural Information Processing Systems. (2018)
5. Tan, H., Bansal, M.: LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. (2019)
6. Andreas, J., Rohrbach, M., Darrell, T., Klein, D.: Neural Module Networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016)
7. Hu, R., Andreas, J., Rohrbach, M., Darrell, T., Saenko, K.: Learning to Reason: End-to-End Module Networks for Visual Question Answering. In: IEEE International Conference on Computer Vision (ICCV). (2017)
8. Hu, R., Andreas, J., Darrell, T., Saenko, K.: Explainable Neural Computation via Stack Neural Module Networks. In: Computer Vision - ECCV. (2018)
9. Yi, K., Wu, J., Gan, C., Torralba, A., Pushmeet, K., Tenenbaum, J.B.: Neural-Symbolic VQA: Disentangling Reasoning from Vision and Language Understanding. In: Advances in Neural Information Processing Systems. (2018)
10. Mao, J., Gan, C., Deepmind, P.K., Tenenbaum, J.B., Wu, J.: The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision. In: International Conference on Learning Representations (ICLR). (2019)
11. Yu, L., Chen, X., Gkioxari, G., Bansal, M., Berg, T.L., Batra, D.: Multi-Target Embodied Question Answering. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2019)
12. Roh, J., Paxton, C., Pronobis, A., Farhadi, A., Fox, D.: Conditional Driving from Natural Language Instructions. In: Conference on Robot Learning. (2019)
13. Wang, X., Huang, Q., Celikyilmaz, A., Gao, J., Shen, D., Wang, Y.F., Wang, W.Y., Zhang, L.: Reinforced Cross-Modal Matching and Self-Supervised Imitation Learning for Vision-Language Navigation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2019)
14. Balajee Vasudevan, A., Dai, D., Van Gool, L.: Talk2Nav: Long-Range Vision-and-Language Navigation with Dual Attention and Spatial Memory. International Journal of Computer Vision (2020)
15. Feng, W., Zhuo, H.H., Kambhampati, S.: Extracting Action Sequences from Texts Based on Deep Reinforcement Learning. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI). (2018)
16. Nyga, D., Roy, S., Paul, R., Park, D., Pomarlan, M., Beetz, M., Roy, N.: Grounding Robot Plans from Natural Language Instructions with Incomplete World Knowledge. In: 2nd Conference on Robot Learning. (2018)
17. Zhang, H., Lai, P.J., Paul, S., Kothawade, S., Nikolaidis, S.: Learning Collaborative Action Plans from YouTube Videos. In: International Symposium on Robotics Research (ISRR). (2019)

18. Johnson, J., Fei-Fei, L., Hariharan, B., Zitnick, C.L., Van Der Maaten, L., Girshick, R.: CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017)
19. Nazarczuk, M., Mikolajczyk, K.: SHOP-VRB: A Visual Reasoning Benchmark for Object Perception. In: International Conference on Robotics and Automation (ICRA). (2020)
20. Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., De, D., Casas, L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., Riedmiller, M.: DeepMind Control Suite. CoRR (2018)
21. Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., Fox, D.: ALFRED A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2020)
22. Mandlekar, A., Booher, J., Spero, M., Tung, A., Gupta, A., Zhu, Y., Garg, A., Savarese, S., Fei-Fei, L.: Scaling Robot Supervision to Hundreds of Hours with RoboTurk: Robotic Manipulation Dataset through Human Reasoning and Dexterity. In: International Conference on Intelligent Robots and Systems, (IROS). (2019)
23. Fan, L., Zhu, Y., Zhu, J., Liu, Z., Zeng, O., Gupta, A., Creus-Costa, J., Savarese, S., Fei-Fei, L.: SURREAL: Open-Source Reinforcement Learning Framework and Robot Manipulation Benchmark. In: Conference on Robot Learning. (2018)
24. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: International Conference on Computer Vision (ICCV). (2017)
25. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016)
26. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to Sequence Learning with Neural Networks. In: Advances in Neural Information Processing Systems. (2014)
27. Hochreiter, S., Schmidhuber, J.: Long Short-term Memory. Neural computation **9** (1997) 1735–1780
28. Bahdanau, D., Cho, K., Bengio, Y.: Neural Machine Translation by Jointly Learning to Align and Translate. In: International Conference on Learning Representations (ICLR). (2015)
29. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning **8** (1992) 229–256
30. Kingma, D.P., Lei Ba, J.: Adam: A Method for Stochastic Optimization. In: International Conference on Learning Representations (ICLR). (2015)