Supplementary Material for Scale-Localized Abstract Reasoning

A. An example of biases in the construction of the negative choices

In the main paper, Sec. 2.1, we discuss the importance of the selected approach for the creation of the negative choices and that it should be done carefully. In this section, we elaborate on this topic by showing examples for bad edge cases.

Consider the following four sets of answers to the question "when and where was 'The Declaration of the Rights of Man' published?"

1. (a) Aug. 1789, France (b) Albert Einstein (c) 55% (d) Alpha Centauri A
2. (a) Aug. 1789, France (b) Nov. 1784, Italy (c) Dec. 1780, Brazil (d) Feb. 1782, Japan
3. (a) Aug. 1789, France (b) Nov. 1789, France (c) Aug. 1784, France (d) Aug. 1789, Italy
4. (a) Aug. 1789, France (b) Nov. 1789, France (c) Nov. 1784, France (d) Nov. 1784, Italy

In all options, the correct answer is the same (a). However, there is a big difference in the negative examples that make the question either easier of harder to answer.

The first option is too simple. The negative choices do not fit the domain of the question and it can be solved without any prior knowledge (by prior knowledge, we mean having some information about the questions). The only answer that fits the domain is also the correct answer.

In the second option, all negative choices are from the same domain as the correct answer. The negative choices are similar enough so that the correct answer cannot be chosen, beyond luck, without having at least some prior knowledge. However, the options are at the same time random enough so that knowing only one attribute (month, year, location) is sufficient to know the answer. For example, one only needs to know that the declaration was published in France to correctly answer the question, without knowing when it was published.

In the third options, each negative choice is produced by changing a single attribute from the correct answer. Unlike the previous option, knowing only one attribute is not enough to eliminate all negative choices, which makes the question more difficult. However, one could notice that 'Aug', '1789' and 'France' are majority attributes in all the answers. If this pattern is recurring across all questions in the test, the participant can learn how to locate the correct answers without even looking at the question. Paradoxically, this issue becomes more severe when more choices are given, in contrast to the purpose of adding more choices.

Finally, the fourth option overcomes all the limitations of the previous examples. The negative choices are sufficiently related so that elimination requires prior knowledge and the attributes are diverse enough so that a majority guess does not promise a correct answer. However, this option is more complicated to produce and a sophisticated method for generating the negative choices needs to be designed.

Table 1. Dataset benchmarks using ResNet. Accuracy(%)

	PGM			RAVEN			
Model	all	lines	shapes	ORIG	FAIR	R-IN	R-ALL
ResNet-Blind	18.6	33.8	14.7	80.2	17.2	13.1	13.5
ResNet-MC	41.1	60.7	28.7	72.5	24.5	52.8	81.6
ResNet-SC	48.9	67.1	40.1	40.4	58.4	82.9	94.4

A.1. Implications for RAVEN

As explained in the related work (Sec. 2), the negative examples in RAVEN are generated by changing a single arbitrary attribute from the correct answer every time. This approach is equivalent to option (3) in the example given above. The problem with this approach is that the model can learn to extract the properties of each attribute from the choice images and compare which values occur in most options. The model can then make an educated guess on which choice is the correct answer.

In the next section, we show that this topic is not just a hypothetical concern, but an actual flaw of the RAVEN dataset that is easily exploited by MC protocol models.

B. Dataset analysis

In Sec. 2.1 of the paper, we discuss the datasets and their potential pitfalls. We then define two different protocols (SC and MC) that can be followed when challenging this task. In Sec. 3, we provide a new dataset (RAVEN-FAIR)

that is better suited than the original dataset for the MC protocol. We have discussed the results of experiments done on the blind, SC and MC settings. In this section, we provide the details of these experiments and their results.

Before using the PGM and RAVEN datasets in our experiments. We analyzed each of them in how they perform on three tests: (i) context-blind test, (ii) multiple choice test (iii) single choice test. The results can be seen in Tab. 1. The model for each test is based on the ResNet16, with an adjusted input and output layer to comply with each test. For the context-blind test, the model (ResNet-Blind) accepts the eight choices, without the context images, as an 8-channel image and produces a probability for each of the input channels to be the correct answer. For the multiple choice test, the model (ResNet-MC) accepts all 16 images as a 16-channel image (context:1-8, choices:9-16) and produces the probability for channels 9-16, which are the choices. For the single choice test, the model (ResNet-SC) accepts a 9-channel image with the context being the first eight channels and a choice added as the ninth channel. The model is fed eight iterations for each question to evaluate each choice separately, and the image with the highest score is chosen as the correct one.

For PGM, we evaluated not only the full dataset, but also two subsets of it, where rules were only applied on the lines or on the shapes. Each subset accounts for roughly a third of the dataset and the remaining third is where both line and shape rules are applied simultaneously. In Tab. 1, the full dataset came out to be relatively balanced, it had a low score on the context-blind test (18.6%) and balanced score between the MC and SC protocols (41% for MC and 48.9% for SC). We have found the lines subset to be significantly easier overall than the shapes subset across all tests, which suggests that the dataset could be improved in that regard.

For RAVEN, we compared four different versions of the dataset. The difference between the versions is the method in which the negative choices were selected. ORIG is the original version of RAVEN. FAIR is our proposed improved dataset. R-IN and R-ALL are two versions where the negative choices were chosen randomly. In R-IN, the negative examples were sampled from the same domain as the correct answer. In R-ALL, we sampled from all domains. The four versions of the dataset represent the four options presented in the question earlier in Sec. A. R-ALL represents option 1, since the negative choices are out of the domain of the correct answer. R-IN represents option 2, since the negative choices are randomly selected from within the domain of the correct answer. ORIG represents option 3, since the negative choices are one attribute away from the correct one. FAIR represents option 4, since it aims not to be too conditioned on the correct answer while not being too random.

The evaluation in Tab. 1, shows that RAVEN indeed fails the context-blind test (80.2%). On the other hand, R-IN (13.1%) and R-ALL (13.3%) were unsolvable in this setting, as expected. RAVEN-FAIR passed the context-blind as well (17.2%), with an equal performance with PGM. When the context was added, both R-IN and R-ALL became notably easy to solve (82.9%, 94.4% respectively on ResNet-SC), which highlights that sampling negative choices randomly is a bad design. ORIG turned out to be more difficult than FAIR in the SC test. This is reasonable since all negative examples are very close to the correct answer in ORIG, which makes them more challenging than those of FAIR. On the other hand, ORIG turned out to be really easy in the MC test (72.5%), while FAIR was significantly harder (40.4%). This is due to the fact that ORIG fails the context-blind test and is easy to solve when all the choices are presented simultaneously. This evaluation shows that one needs to be very careful in evaluating models with RAVEN, since it cannot be used in the MC setting.

Interestingly, except for the original RAVEN, ResNet-SC performed consistently better than ResNet-MC. This is in contrast to the common sense that MC is a simpler setting than SC due to capability to compare the choices. We conclude that this is an architectural drawback of ResNet-MC that it has less practical capability, since it processes all choices at once without added capacity. It is also not permutation invariant to the choices, which means that changing the order of the choices could lead to a different result.

C. Analyzing PGM per rule and stage

In Sec. 5, we analyze the performance on PGM per rule and per stage in our networks. This allows us to determine which stage is responsible for solving each task. We have shown visually how each stage performs on each task. In addition to that, Tab. 2 shows the accuracy for in total and for each stage with respect to each rule. The table also shows the accuracy per type of operation ('progression', 'logical', 'consistent-union'), where this separation was not done in the main paper. We noticed that the operations have sometimes different performances. For example, in PGM_meta, the rules 'shape-color', shape-type' and 'shape-size' work much better in 'progression' than in the other operations. We also noticed that, for each rule, the same resolution solves all operations.

To better understand how each resolution solves each type of rule, we performed a t-SNE analysis of the embedding layer v_t of each resolution, for the embedding of the correct answer I^{a^*}. The analysis was done on the fully trained model with $\mathcal{L}_3$. It can be seen in Fig. 1. It should be noted that separation of the rules in the latent space is not a requirement for accuracy in solving the task, but we have observed high correlation between the two in each resolution and it is a valuable analysis nevertheless. (a) The upper resolution is good at separating most rule types. It even sep-

Table 2. Understanding the role of each scale in PGM. Accuracy(%)

			Operation											
			Progression				XOR+OR+AND				Consistent Union			
	Subset	Attribute	ALL	H	M	L	ALL	H	M	L	ALL	H	M	L
PGM	Shape	Color	14.8	14.8	12.2	14.1	15.4	14.7	14.9	12.1	17.3	16.2	14.6	11.9
		Type	23.8	20.8	12.6	11.5	12.9	13.5	12.1	10.3	15.7	11.0	12.1	13.2
		Size	96.5	95.7	13.5	11.5	51.0	43.9	13.7	14.9	79.1	73.6	12.0	15.1
		Position	✗				99.9	99.9	12.6	14.7	✗			
		Number	100	12.7	100	12.4	✗				100	13.7	100	13.7
	Line	Color	100	13.3	10.7	100	96.9	12.4	12.6	92.9	99.6	12.5	11.4	99.9
		Type	✗				100	39.1	13.1	30.4	100	100	15.9	12.8
PGM_meta	Shape	Color	71.5	58.9	11.8	15.6	31.4	29.0	12.3	12.0	40.8	28.5	12.7	15.4
		Type	92.6	88.8	13.0	17.5	52.0	47.8	15.2	13.5	65.8	47.7	12.8	12.8
		Size	92.3	91.2	12.7	15.0	56.3	45.6	11.7	12.1	72.5	65.9	15.1	14.3
		Position	✗				99.9	94.6	68.8	12.5	✗			
		Number	100	14.8	100	12.1	✗				99.6	13.7	99.4	14.6
	Line	Color	100	12.3	13.8	100	96.0	11.3	12.4	87.3	100	12.1	10.6	99.9
		Type	✗				100	50.8	12.9	49.2	100	99.9	15.7	23.4
PGM $\mathcal{L}_3$	Shape	Color	91.5	89.5	12.7	13.7	54.5	48.7	12.0	12.7	88.0	83.3	12.2	13.1
		Type	98.2	97.2	14.1	12.1	80.0	76.9	12.6	12.5	91.3	88.5	13.0	11.5
		Size	98.0	92.5	11.7	12.6	79.8	76.4	12.8	12.4	93.9	88.2	12.6	13.9
		Position	✗				99.9	98.5	99.9	12.5	✗			
		Number	100	27.2	100	12.1	✗				100	25.3	99.9	12.5
	Line	Color	100	12.6	99.2	100	97.7	14.6	87.7	97.2	100	12.5	97.9	98.3
		Type	✗				99.9	99.5	99.8	39.3	100	99.6	99.8	34.4

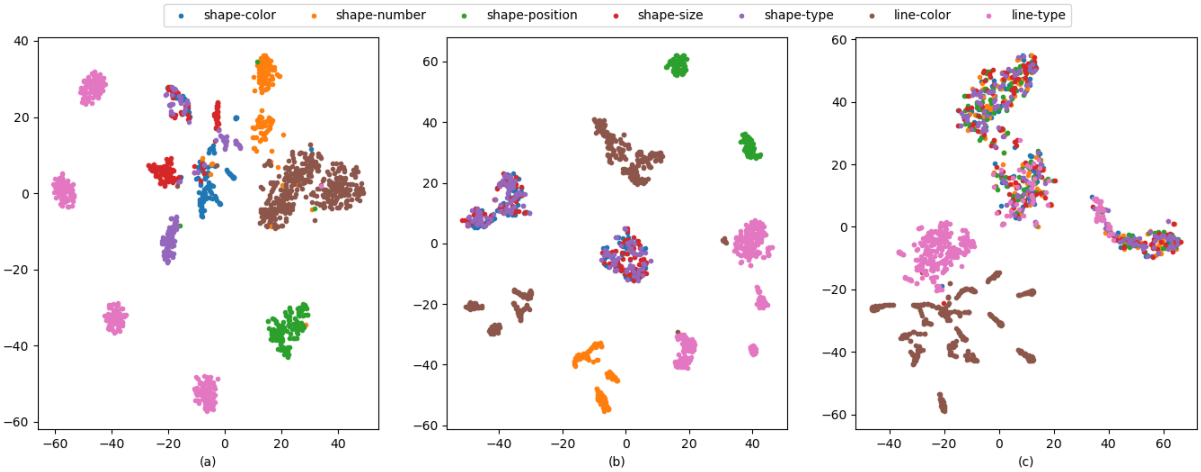
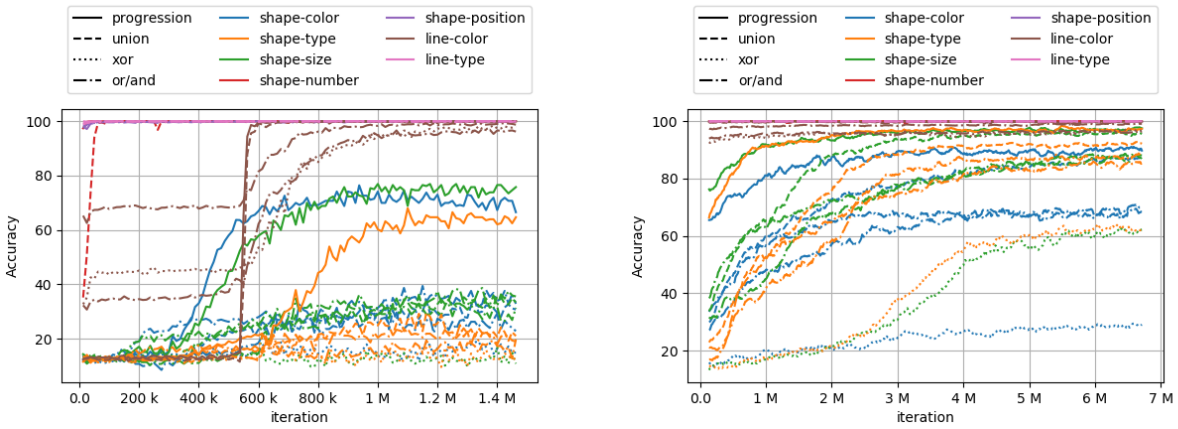


Figure 1. t-SNE analysis of each resolution. (a) High resolution v_h. (b) Middle resolution v_m. (c) Low resolution v_l.

arates 'line-type' into distinct sub-groups, which we have found to be correlated to the relational operation 'progression', 'union', 'xor', etc'. (b) The middle resolution is very good at separating some tasks, such as 'line-type', 'line-color', 'shape-number', and 'shape-position'. These rules are also separated into sub-groups and are also where this resolution has a high accuracy on. The other rules are not separated at all and the resolution also had trouble in solving them. (c) The lower resolution had the lowest overall performance in accuracy and also in separating the rules. However, we did notice that it was exceptional in separating the 'line-color' rules into multiple sub-groups. The lower reso-



(a) (b)

Figure 2. Accuracy for each rule over time. (a) Without $\mathcal{L}_3$. (b) With $\mathcal{L}_3$. It can be seen that during each 'step', the model is focused at learning a different subset of tasks. It is also noticeable that the added multihead loss immediately improves the rules that the model was not able to solve.

lution also has the highest accuracy on this type of rules.

Aside from the final results per rule, we noticed an interesting behaviour in the training convergence. Fig. 2 shows the accuracy of each rule and operation over time. We noticed that the model does not learn all the rules at the same time. Instead, the training appears to go in stages, where the model learns a particular kind of rule at each stage. In Fig. 2(a), we show the learning progress without $\mathcal{L}_3$, and in Fig. 2(b), we show the contribution of $\mathcal{L}_3$ when it is added. It was especially surprising to see that the model only started to learn 'line-color' very late (around 500K iterations) but did it very fast (within 50K iterations for 'progression' and 'union'). The model is able to learn numerous rules without the multihead loss $\mathcal{L}_3$. However, many rules, especially in the 'shape' category are not solved well. The addition of the multihead loss immediately and significantly increases the performance on all rules.

Since training took a long time and we measured only after each epoch, the plot doesn't show the progress in the early stages of the training, which would show when the 'easier' tasks have been learned. Future work can focus on this 'over-time' analysis and try to explain: (i) 'why are some rules learned and others not?', (ii) 'why are some rules learned faster than others?', (iii) 'how can the training time be shortened?', (iv) 'would the rules that were learned late also be learned if the easier rules were not present?'.

D. Architecture of each sub-module

We detail each sub-module used in our method in Tab. 4-7. Since some modules re-use the same blocks, Tab. 3 details a set of general modules.

Table 3. General modules, with variable number of channels c.

Module	layers	parameters	input	output
ResBlock3(c)	Conv2D	CcK3S1P1	x	
	BatchNorm			
	ReLU			
	Conv2D	CcK3S1P1		
	BatchNorm			x'
	Residual		(x, x')	$x'' = x + x'$
	ReLU			
DResBlock3(c)	Conv2D	CcK3S1P1	x	
	BatchNorm			
	ReLU			
	Conv2D	CcK3S1P1		
	BatchNorm			x'
	Conv2D	CcK1S2P0	x	
	BatchNorm			x_d
	Residual		(x_d, x')	$x'' = x_d + x'$
	ReLU			
ResBlock1(c)	Conv2D	CcK1S1P0	x	
	BatchNorm			
	ReLU			
	Conv2D	CcK1S1P0		
	BatchNorm			x'
	Residual		(x, x')	$x'' = x + x'$
	ReLU			

Table 4. Encoders E_h, E_m, E_l

Module	layers	parameters	input	output
E_h	Conv2D	C32K7S2P3	I^1	
	BatchNorm			
	ReLU			
	Conv2D	C64K3S2P1		
	BatchNorm			
	ReLU			e_h^1
E_m	Conv2D	C64K3S2P1	e_h^1	
	BatchNorm			
	ReLU			
	Conv2D	C128K3S2P1		
	BatchNorm			
	ReLU			e_m^1
E_l	Conv2D	C128K3S2P1	e_m^1	
	BatchNorm			
	ReLU			
	Conv2D	C256K3S2P1		
	BatchNorm			
	ReLU			e_l^1

Table 5. Relation networks RN_h, RN_m, RN_l

Module	layers	parameters	input	output
RN_h	Conv2D	C64K3S1P1	(e_h^1, e_h^2, e_h^3)	
	ResBlock3	C64		
	ResBlock3	C64		
	Conv2D	C64K3S1P1		
	BatchNorm			r_h^1
RN_m	Conv2D	C128K3S1P1	(e_m^1, e_m^2, e_m^3)	
	ResBlock3	C128		
	ResBlock3	C128		
	Conv2D	C128K3S1P1		
	BatchNorm			r_m^1
RN_l	Conv2D	C256K1S1P0	(e_l^1, e_l^2, e_l^3)	
	ResBlock1	C256		
	ResBlock1	C256		
	Conv2D	C256K1S1P0		
	BatchNorm			r_l^1

Table 6. Bottlenecks B_h, B_m, B_l

Module	layers	parameters	input	output
B_h	DResBlock3	C128	b_h	
	DResBlock3	C128		
	AvgPool2D			v_h
B_m	DResBlock3	256	b_m	
	DResBlock3	C128		
	AvgPool2D			v_m
B_l	Conv2D	C256K1S1P0	b_l	
	BatchNorm			
	ReLU			
	ResBlock1	C128		v_l

Table 7. MLP

Module	layers	parameters	input	output
MLP	Linear	C256	(v_h, v_m, v_l)	
	BatchNorm			
	ReLU			
	Linear	C128		
	BatchNorm			
	ReLU			
	Linear	C1		
	Sigmoid			$p(y = 1\|I^a, I_C)$