Appendix A Training Data Sampling

The basic sampling unit of training data is triplet as (q, $\{img\}$, $\{ans\}$). Generally, as shown in Fig.1, we maintain three sets $\{Unmet\}$, $\{Unsolved\}$, $\{Solved\}$ to distinguish training data in different status.

Intuitively, $\{Unmet\}$ contains training data that have not been met and used. At the beginning of learning, all the training data is stored in $\{Unmet\}$.

$\{Unsolved\}$ contains training data that has been sampled from $\{Unmet\}$ but on which the final accuracy achieved in the following search did not reach the value of a hyperparameter named *Acceptable_Boundary*.

$\{Solved\}$ contains training data that has been sampled from $\{Unmet\}$ or $\{Unsolved\}$ and the final accuracy reached the *Acceptable_Boundary*.

We denote the numbers of training data triplets in these three sets as N_{um}, N_{us} and N_s, respectively.

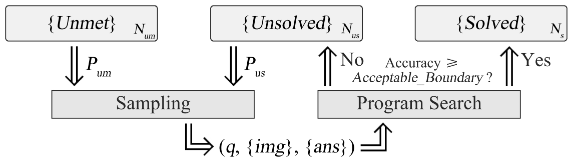

Fig. 1. Data sampling strategy

For each *Sample* step in each training loop, the training data can be sampled from either $\{Unmet\}$ or $\{Unsolved\}$ with probabilities P_{um} and P_{us}. These two probabilities are calculated through Equation 1.

$$P_{um} = \begin{cases} e^{-\frac{N_{us}}{N_s+1}}, & \text{if } N_{um} > 0 \; ; \\ 0, & \text{otherwise} \end{cases} \tag{1a}$$

$$P_{us} = 1 - P_{um} \tag{1b}$$

Appendix B Program Graph Initialization

To initialize the Program Graph, at most three initial program nodes are created as the starting points for the following search. The programs of them are:

i) The program predicted by the Program Predictor model.

ii) The program found for the question within $\{Solved\}$ that is closest to the current given question in the sense of semantics.

iii) The shortest legal program.

Specifically for ii), this term only works when $\{Solved\}$ is not empty. If so, a pre-trained sentence embedding model $SE(\cdot)$ is utilized to judge the semantic distance between questions and find the question q_c from $\{Solved\}$ that is semantically closest to the current given question q. This process can be expressed as Equation 2. Here, $SE(\cdot)$ takes question sentence as input and outputs a fixed-length vector. $\parallel SE(q) - SE(q_s) \parallel$ judges the L^2 distance between $SE(q)$ and $SE(q_s)$. Then, the program that has been found for q_c in previous searches becomes the program used in term ii) to initialize the Program Graph.

$$q_c = \underset{q_s \in \{Solved\}}{\arg\min} \parallel SE(q) - SE(q_s) \parallel \tag{2}$$

Appendix C Legality Check for Programs

As stated in Section 3.3, our rules for generating mutations on programs can ensure their legality of structure, but not necessarily their legality of semantics. Here, the illegality of semantics mainly comes from the type system of modules. Within NMN, the inputs and outputs passed between modules are restricted with types such as feature map, number, object, or set of objects. The calculation of NMN fails if the intermediate data fed to a module does not match the data type that module requires.

Generally, there are two solutions to this problem. One is to add the illegal programs to the Program Graph anyway yet mark these programs as non-executable and skip the step of trying these programs to get the accuracies. However, excessive illegal programs within the Program Graph waste plenty of searching steps on them so that the efficiency of search drops obviously.

The other solution is to simply refuse to add these illegal programs to the Program Graph. However, in this way the Program Graph is possible to become disconnected. Therefore, some sub-graphs may never be reached from others.

In consideration of this, we applied a compromise between these two solutions. We used a hyperparameter named $Tolerance$ to restrict the maximum count of data type mismatches that can be tolerated. The programs of which the count of data type mismatches is not greater than $Tolerance$ will still be added to the Program Graph although they cannot be executed to obtain the accuracy. This setting can balance the efficiency and coverage of the search.

Appendix D Modules Used in FigureQA Dataset

The modules used in the experiment on the FigureQA dataset are shown in Table 1. Here, the column of "Shape" indicates the number and type of inputs and output. The column of "Architecture" indicates whether the module is predefined with rule-based calculation, or is a trainable neural network.

Specifically for the behavior of each module, "Find Element" finds an element that matches the given keyword from all the detected elements. Here, keywords are the name of colors extracted from the questions. Because there are at most

Table 1. Modules used in the experiment on FigureQA dataset

Name	Shape	Architecture	Number
Find Element	(None) $\rightarrow$ Element	pre-defined	2
Look Up	(Element) $\rightarrow$ Element	pre-defined	1
Look Down	(Element) $\rightarrow$ Element	pre-defined	1
Look Left	(Element) $\rightarrow$ Element	pre-defined	1
Look Right	(Element) $\rightarrow$ Element	pre-defined	1
Find Same	(Element) $\rightarrow$ Elements	pre-defined	1
Discriminator	(Element/Elements/None) * 2 $\rightarrow$ Answer	neural network	N

two keywords within a question, two of this module are required and each of them corresponds to one of the keywords.

"Look Up" finds the closest element that is in the area of from 45° top left to 45° top right of the given element.

"Look Down", "Look Left", and "Look Right" behave similarly to "Look Up".

"Find Same" finds a set of elements with the same attributes as the given element. In this experiment, we specify this attribute to color.

"Discriminator" has two inputs. For each input, it masks the original image with the bounding boxes of the given element or sets of elements. Then, the masked image is fed to a neural network to infer the answer. The input can also be empty. In this case, it would directly feed the original image to the neural network. To compare our method with existing work fairly, we use a 40-layer DenseNet similar to the one applied in PReFIL as the backbone of the Discriminator. The architecture of Discriminator is shown in Fig.2. The number of filters in the first convolutional layer of DenseNet is 64. Considering that the two inputs of Discriminator are parallel and most of their features are similar, the first convolutional layer works on them independently with shared weight. All three following dense blocks have 12 layers. Their growth rate is set to 12. The number of final classes is 2 representing the answer "Yes" or "No" in FigureQA.

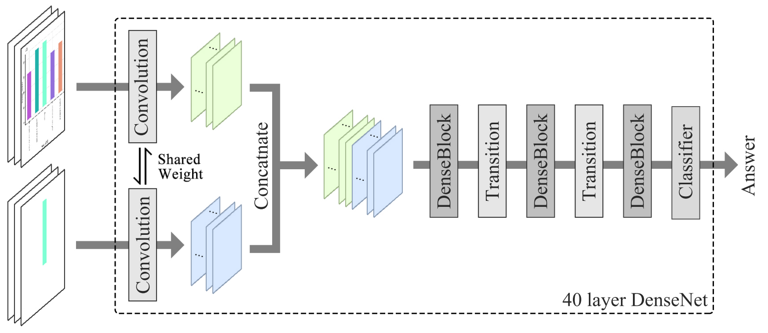

Fig. 2. Architecture of our Discriminator with a 40 layer DenseNet as backbone

For training, we used cross-entropy loss and SGD optimizer with learning rate decay. The batch size is set to 64. The learning rate is initialized to be 0.1 and drops to 0.01, 0.001, 0.0001, and 0.00001 on epoch 8, 12, 16, and 20, respectively. The maximum number of the epochs of training is 24, yet considering that the training of a 40-layer DenseNet on the entire 24 epochs is highly time-consuming, during the search only the training on the first 4 epochs are conducted and the validation accuracy is returned then. After the search on each question is completed, the Discriminator specified by the optimal program will be trained again on the entire 24 epochs.

Appendix E Results by Question Type and Figure Type in FigureQA Dataset

Table 2. Accuracy on Test Set 2 by different question types.

Question Template	RN	Human	PReFIL	Ours
Is X the minimum?	76.78	97.06	97.20	**98.44**
Is X the maximum?	83.47	97.18	98.07	**98.79**
Is X the low median?	66.69	86.39	93.07	**94.07**
Is X the high median?	66.50	86.91	93.00	**94.29**
Is X less than Y?	80.49	96.15	98.20	**99.43**
Is X greater than Y?	81.00	96.15	98.07	**99.45**
Does X have the minimum area under the curve?	69.57	94.22	94.00	**95.77**
Does X have the maximum area under the curve?	78.45	95.36	96.91	**97.65**
Is X the smoothest?	58.57	78.02	71.87	**80.90**
Is X the roughest?	56.28	79.52	74.67	**85.19**
Does X have the lowest value?	69.65	90.33	92.17	**95.42**
Does X have the highest value?	76.23	93.11	94.83	**96.68**
Is X less than Y?	67.75	90.12	92.38	**95.19**
Is X greater than Y?	67.12	89.88	92.00	**95.29**
Does X intersect Y?	68.75	89.62	91.25	**95.22**
Overall	72.18	91.21	92.79	**95.28**

Table 3. Accuracy on Test Set 2 by different figure types.

Figure Type	RN	Human	PReFIL	Ours
Vertical Bar	77.13	95.90	98.25	**98.55**
Horizontal Bar	77.02	96.03	97.98	**99.32**
Pie	73.26	88.26	92.84	**94.31**
Line	66.69	90.55	87.79	**92.66**
Dot Line	69.22	87.20	89.57	**93.11**
Overall	72.18	91.21	92.79	**95.28**