Simple Book Example

TeXstudio Team

December 19, 2022

Contents

Chapter 1

The First Chapter

1.1 Main Task

Es müssen mathematische und algorithmische Grundlagen für neurosymbolische KI-Systeme entwickelt werden. Ein ganzheitlicher Ansatz sollte die Perspektive der Verwendung verallgemeinerbarer und robuster neuronaler Netzmodelle mit der Ausdrucks- und Argumentationskraft symbolischer Systeme im Entwurfsprozess verbinden. Zu diesem Zweck sollen neuartige neurosymbolische Modelle und dynamische Architekturen entwickelt werden, um in Gegenwart von strukturierten, verrauschten und heterogenen Daten effektiv zu lernen. The goal of neural-symbolic AI(NeSy) is to integrate symbolic reasoning and neural networks, where the first topic focus on the task of **reasoning** and the second topic focus on the task of **learning**. NeSy is different from statistical relational learning and artificial intelligence (StarIAI), which is also a domain that focus on the integration of learning and reasoning. [8] talked about neural symbolic AI. Image question answering often requires multiple steps of reasoning. [10]

1.2 Related Work

[11] injects discrete logical constraints(DLC) into neural network. It represents the DLC as a loss function, then use a mechanism called *Straight-Through-Estimator(STE)* to update the neural network's weight in the direction that the binarized outputs satisfy the logical constraints. They argued that this method is better than the others that require heavy symbolic computation for computing gradients.

Chapter 2

Dataset

2.1 CLEVR

CLEVR[5] is a dataset for *image reasoning.* CLEVRER is a dataset for *video reasoning.*

Objects and Relationships

- Three object shapes: cube, sphere, and cylinder.
- Two absolute sizes: small and large.
- Two materials: shiny and matte.
- Eight colors.
- Four relationships: left, right, behind, in front

Scene representation A scene can be represented by a scene graph.

Image generation Images are generated by randomly sampling a scene graph and render it using Blender.

Question representation Each question is associated with a *functional program* that can be executed on an image's scene graph.

Functional programs are built from simple basic building blocks, which are elementary operations of visual reasoning, such as querying object attributes, counting sets of objects, or comparing values.

Question size is defined as the number of functions in its program. Effective question is defined as the smallest equivalent program. Effective size is the size of effective question. Large effective size leads to long reasoning chains, which gives a higher error rate.

Complex questions can be represented by compositions of simple building blocks.

Question families A question family contains a template for constructing functional programs and several text templates providing multiple ways of expressing these programs in natural language.

Question generation

1. choose a question family (depth-first search)
2. select values for each of its template parameters
3. execute the resulting program on the image's scene graph to find the answer
4. use one of the text templates from the question family to generate the final natural-language question

Question topology

1. chain-structured, *what color is the cube to the right of the yellow sphere.*
2. tree-structured, maybe more difficult, require models to perform two sub-tasks in parallel before fusing their results. *How many cylinders are in front of the small thing and on the left side of the green object.*

2.2 DAQUAR

This dataset is proposed in [6].

- 6795 training questions on 795 images
- 5673 test questions on 654 images
- question types
  - object
  - color
  - number

2.3 COCO-QA

This dataset is proposed in [7].

- 78736 training questions on 8000 images
- 38948 test questions on 4000 images
- question types
  - object
  - color
  - number
  - location

2.4 VQA

This dataset is proposed in [1].

- 0.25M images
- 0.76M questions
- 10M answers

2.5 Image Net

[4]

Chapter 3

Language Models

3.1 LSTM

LSTM is used for question analyzing tasks. It can reserve the state of a sequence.

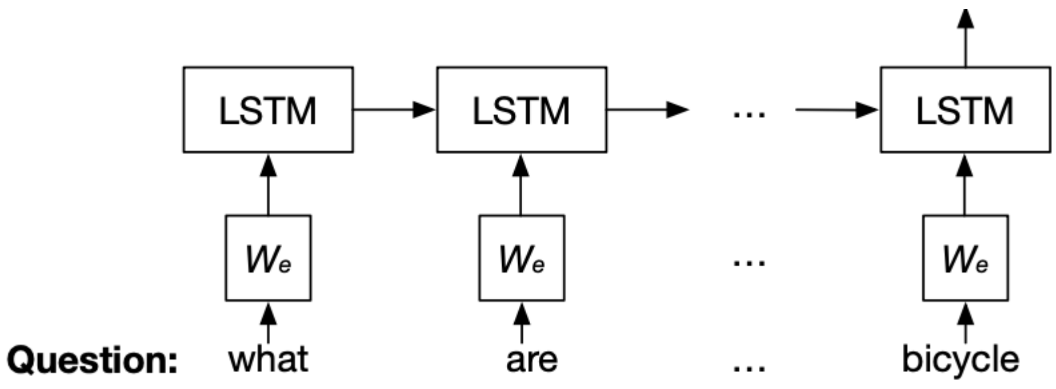

Figure 3.1: LSTM based question model *what are sitting in the basket on a bicycle.* [10]

Given the question

$$q = [q_1, ..., q_T]$$

where q_t is the one hot vector representation of word at position t. We embedded the words to a vector space through an embedding matrix and get the input vector x_t,

$$x_t = W_e q_t, t \in \{1, 2, ..., T\}$$

Then for every time step, we feed the embedding vector of words in the question to LSTM, then output a hidden state h_t,

$$h_t = LSTM(x_t), t \in \{1, 2, ..., T\}$$

The LSTM update process uses the gate mechanism. An input gate i_t controls how much the current input x_t updates the memory cell.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

The forget gate f_t controls how much information from past state c_{t-1} is preserved.

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

The output gate o_t controls how much information of the memory is fed to the output as hidden state

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

The memory cell c_t reserves the state of a sequence

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

The output hidden state h_t is updated as

$$h_t = o_t \tanh(c_t)$$

The final hidden layer is taken as the representation vector for the question, i.e.

$$v_Q = h_T$$

3.2 CNN

Similar to LSTM based model, CNN model embeds words to **word vectors**

$$x_t = W_e q_t$$

Then it concatenates the word vectors to **question vectors**

$$x_{1:T} = [x_1, x_2, ..., x_T]$$

Let c denotes the size of convolution filters, t denotes the t-th convolution output, W_c is the convolution weight, b_c is the bias. The convolution output is given by

$$h_{c,t} = \tanh(W_c x_{t:t+c-1} + b_c)$$

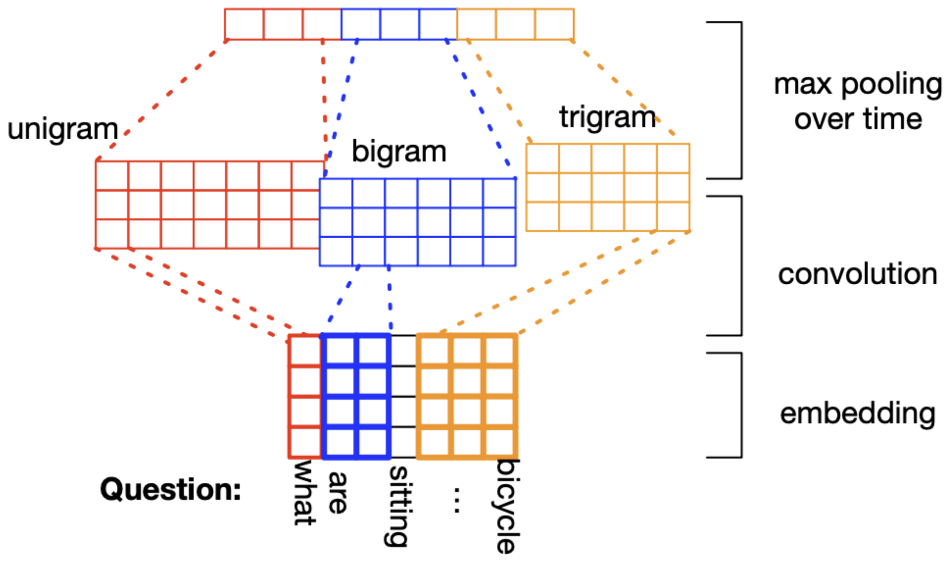

Figure 3.2: CNN based question model *what are sitting in the basket on a bicycle.* [10]

The **feature map** with convolution filter size c is given by

$$h_c = [h_{c,1}, h_{c,2}, ..., h_{c,T-c+1}]$$

Then we apply max-pooling over the feature maps of the convolution size c and denote it as

$$\tilde{h}_c = \max_t [h_{c,1}, h_{c,2}, ..., h_{c,T-c+1}]$$

For convolution feature maps of different sizes c, we concatenate them to form the feature representation vector of the whole question sentence

$$h = [\tilde{h}_1, \tilde{h}_2, \tilde{h}_3]$$

Hence the CNN based question vector is

$$v_Q = h$$

3.3 Stacked Attention Networks (SAN)

Let $v_I \in \mathbb{R}^{d \times m}$ denotes the image feature matrix, m is the number of image regions. $v_Q \in \mathbb{R}^d$ is a d dimensional vector. denotes the question feature vector. Let $W_{I,A}, W_{Q,A} \in \mathbb{R}^{k \times d}$, $\oplus$ denotes the addition of a matrix and a vector, which is performed by adding each column of the matrix by the vector. Then the two vectors first go through a single layer neural network

$$h_A = \tanh(W_{I,A}v_I \oplus (W_{Q,A}v_Q + b_A))$$

Let $W_P \in \mathbb{R}^{1\times k}$, then a softmax function is used to generate the attention distribution $p_I \in \mathbb{R}^m$ over the regions of the image, which corresponds to the attention probability of each image region given v_Q.

$$p_I = \text{softmax}(W_P h_A + b_P)$$

Based on the attention distribution we calculate the weighted sum of the image vectors $\tilde{v}_i$.

$$\tilde{v}_I = \sum_i p_i v_i$$

Then $\tilde{v}_i$ is combined with the question vector v_Q to form a refined **query vector** u.

$$u = \tilde{v}_I + v_Q$$

The k-th attention layer is calculated as

$$\begin{aligned} h_A^k &= \tanh(W_{I,A}^k v_I \oplus (W_{Q,A}^k u^{k-1} + b_A^k)) \\ p_I^k &= \text{softmax}(W_P^k h_A^k + b_P^k) \end{aligned} \tag{3.1}$$

where u^0 is initialized to be v_Q. The aggregated image feature vector is added to the previous query vector to form a new query vector:

$$\begin{aligned} \tilde{v}_I^k &= \sum_i p_i^k v_i \\ u^k &= \tilde{v}_I^k + u^{k-1} \end{aligned} \tag{3.2}$$

It will be repeat for K times and then use the final u^K to infer the answer:

$$p_{ans} = \text{softmax}(W_u u^K + b_u)$$

3.4 Convolutional bottleneck attention module (CBAM)

The CBAM[9] module is designed to learn what and where to emphasize or suppress and refines intermediate features effectively.

Given an intermediate feature map $\mathbf{F} \in \mathbb{R}^{C\times H\times W}$ as input, CBAM sequentially infers a 1D channel attention map $\mathbf{M}_c \in \mathbb{R}^{C\times 1\times 1}$ and a 2D spatial

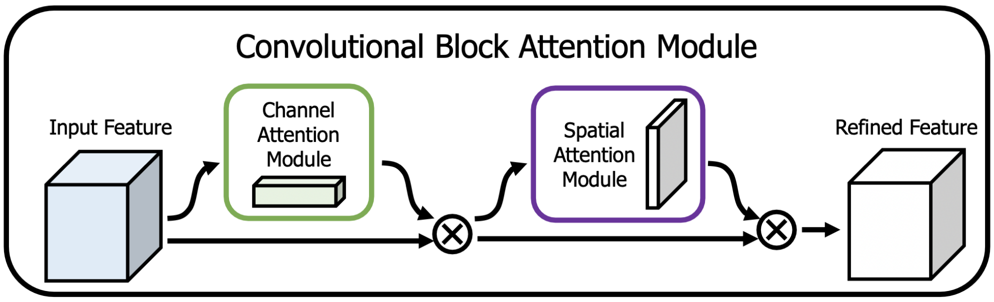

Figure 3.3: The overview of CBAM [9]

attention map $\mathbf{M}_s \in \mathbb{R}^{1\times H\times W}$ as illustrated in Figure 3.3. The overall attention process can be summarized as

$$\begin{aligned}\mathbf{F}' &= \mathbf{M}_c(\mathbf{F}) \otimes \mathbf{F}\\ \mathbf{F}'' &= \mathbf{M}_s(\mathbf{F}') \otimes \mathbf{F}'\end{aligned} \tag{3.3}$$

where $\otimes$ denotes element-wise multiplication.

3.4.1 Channel attention module

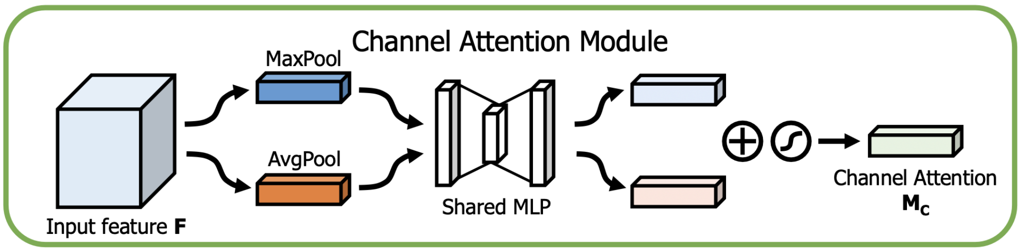

Figure 3.4: Diagram of channel attention module.[9]

The channel attention module focus on **what** is meaningful given an input image. To compute it, we first squeeze the spatial dimension of the input feature map. [13] suggest to use average-pooling to learn the extent of the target object. [9] argues that max-pooling can be used to gather distinctive object features. Thus paper [9] exploits both average-pooling and max-pooling operations to aggregate spatial information. $\mathbf{F}^c_{avg}$ denotes average-pooled features and $\mathbf{F}^c_{max}$ denotes max-pooled features. They are forwarded to a shared network (multi-layer perceptron (MLP)) to produce channel attention map $\mathbf{M}_c \in \mathbb{R}^{C\times 1\times 1}$. MLP has one hidden layer with hidden activation size $\mathbb{R}^{C/r\times 1\times 1}$, where r is the reduction ratio. Formally, the channel attention is computed as

$$\begin{aligned}\mathbf{M}_c(\mathbf{F}) &= \sigma(MLP(AvgPool(\mathbf{F})) + MLP(MaxPool(\mathbf{F}))) \\ &= \sigma(\mathbf{W}_1(\mathbf{W}_0(\mathbf{F}^c_{avg})) + \mathbf{W}_1(\mathbf{W}_0(\mathbf{F}^c_{max})))\end{aligned} \tag{3.4}$$

3.4.2 Spatial attention module

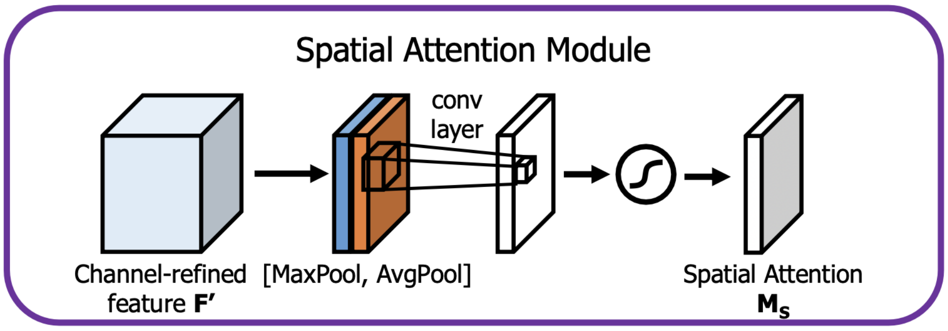

Figure 3.5: Diagram of spatial attention module.[9]

The spatial attention focuses on **where** is an informative part. [12] shows that applying pooling operations along the channel axis is effective in highlighting informative regions. The spatial attention is computed as

$$\begin{aligned}\mathbf{M}_s(\mathbf{F}) &= \sigma(f^{7\times 7}([AvgPool(\mathbf{F}); MaxPool(\mathbf{F})])) \\ &= \sigma(f^{7\times 7}([\mathbf{F}^s_{avg}; \mathbf{F}^s_{max}]))\end{aligned} \tag{3.5}$$

where σ denotes the sigmoid function and $f^{7\times 7}$ represents a convolution operation with the filter size 7×7.

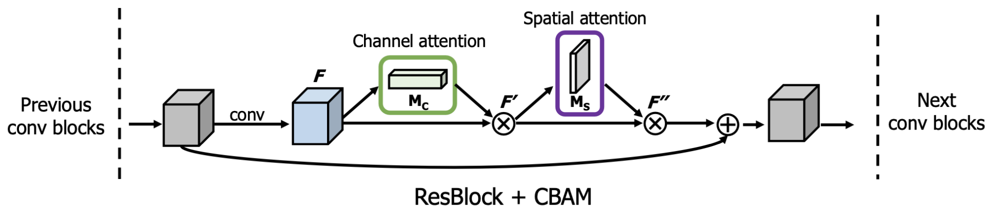

Figure 3.6: CBAM integrated with a ResBlock in ResNet.[9]

3.5 Binarized Neural Network

Let x^b be the binarized variable and x be the real-valued variable.. Deterministic binarization function

$$x^b = Sign(x) = \begin{cases} +1 & if\ x \geq 0 \\ -1 & otherwise \end{cases} \tag{3.6}$$

Stochastic binarization function

$$x^b = \begin{cases} +1 & \text{with probability } p = \sigma(x) \\ -1 & \text{with probability } 1 - p \end{cases} \tag{3.7}$$

where σ is the *hard sigmoid* function

$$\sigma(x) = \text{clip}(\frac{x+1}{2}, 0, 1) = \max(0, \min(1, \frac{x+1}{2}))$$

[3]

Chapter 4

Evaluation

4.1 Metrics

4.1.1 MIN

For the open-ended task, [1] uses following metrics

$$\text{accuracy} = \min\left(\frac{\#\text{ humans that provided that answer}}{3}, 1\right)$$

Thus if at least 3 workers provided the answer, then the answer is 100% accurate.

4.2 Baselines

4.2.1 per Q-type prior

For the open-ended task, pick the most popular answer per question type. [1]

4.2.2 nearest neighbor

Given a test image, question pair, first find the K nearest neighbor questions and associated images from the training set. Then, for the open-ended task, pick the most frequent ground truth answer from this set of nearest neighbor question, image pairs.

Chapter 5

Methods

5.1 Straight-Through-Estimators

Straight-Through-Estimator (STE)[2] is used to estimate the gradients of a discrete function. The problem of discrete function is that its gradient is zero. An STE estimates (or in another word, substitute) the gradient of a function to other values. For example[1] , define a discrete function

$$f(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{5.1}$$

The derivative of $f(x)$ is almost zero everywhere. Let's define an STE as

$$i(x) = x$$

Then we use the derivative of $i(x)$ as a substitution of the derivative of $f(x)$. Thus the gradient of this discrete function $f(x)$ is *not* zero anymore, but equal to $\frac{\partial i(x)}{\partial x} = 1$ everywhere.

5.2 Rule Search Algorithm

[1]https://www.hassanaskary.com/python/pytorch/deep%20learning/2020/09/19/intuitive-explanation-of-straight-through-estimators.html

Algorithm 1 Rule Search Algorithm

```
 1: RulesSet ← $\phi$
 2: PropExistRules ← $\phi$
 3: VerticalPropCombRules ← $\phi$
 4: HorizontalPropCombRules ← $\phi$
 5: for propertyType $A_i$ in $A$ do
 6:     for Object $O_j$ in Image 1 do
 7:         for propertyInstance $b$ in $a$ do
 8:             if PropertyInstance[$A_i$, $O_j$] exists in all images then
 9:                 PropertyInstance[$A_i$, $O_j$].common ← True
10:                 PropExistRules[$A_i$].append(PropertyInstance[$A_i$, $O_j$])
11:             else
12:                 PropertyInstance[$A_i$, $O_j$].common ← False
13:             end if
14:         end for
15:         for VertialComb(subset) $v_k$ in PropExistRules[i] do
16:             if $c_k$ exist in all images then
17:                 VerticalPropCombRules[$A_i$].append($v_k$)
18:             end if
19:         end for
20:     end for
21: end for
22: for Object $O_j$ in image 1 do
23:     for HorizontalComb(subset) $h_k$ in ExistProps of $O_j$ do
24:         if $h_k$ exist in all images then
25:             HorizontalPropCombRules[$O_i$].append($h_k$)
26:         end if
27:     end for
28: end for
```

Algorithm 2 Rule Search Algorithm

```
1: CommonSet ← $\phi$
2: Rules ← $\phi$
3: $N$ images, $M$ properties, image $p$ has $O_p$ objects
4: $A^{O_i \times M}$ denotes property matrix of image $i$.
5: for each element $p$ in $A^{O_i \times M}$ do
6:     if If CommonExist($p$) $==$ $True$ then
7:         $p.common \leftarrow True$
8:         CommonSet.append($p$)
9:     else
10:        $p.common \leftarrow False$
11:    end if
12: end for
13: for a in range (1, CommonSet.length) do
    SubCommonSet ← SubSet(CommonSet, a)
14:    for element $s$ in SubCommonSet do
15:        result ← check(s)
16:        if result == true then
17:            Rules.append(s)
18:        end if
19:    end for
20: end for
```

Algorithm 3 CommonExist Algorithm

```
1: Description: Check if input property exist in every images
2: Input: property P
3: for each image i do
4:     if P exists in image i then
5:         return True
6:     else
7:         return False
8:     end if
9: end for
```

Algorithm 4 Check Algorithm

```
1: Description: Check if input exist in all images
2: Input: property set P := $p_1, p_2, ..., p_k$
3: CommonObjs O $\leftarrow \phi$
4: Candidates $\leftarrow \phi$ For $o_j$ in image 1: $Candidate_Obj \leftarrow \phi$ For each property
   $p$ of object $o_j$ If $p \in P$ type = $p_i.type$ $Candidate_Obj.append(p)$ $o_j.p$ =
   true Else O.append($p_i.obj$)
       For each image i For each $Candidate_Obj$ If $Candidate_Obj$ not in
   $image_i.objs$ return False return True
```

Bibliography

[1] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Batra, and D. Parikh. Vqa: Visual question answering, 2015. URL `https://arxiv.org/abs/1505.00468`.

[2] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. URL `https://arxiv.org/abs/1308.3432`.

[3] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1, 2016. URL `https://arxiv.org/abs/1602.02830`.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[5] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning, 2016. URL `https://arxiv.org/abs/1612.06890`.

[6] M. Malinowski and M. Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input, 2014. URL `https://arxiv.org/abs/1410.0210`.

[7] M. Ren, R. Kiros, and R. Zemel. Exploring models and data for image question answering, 2015. URL `https://arxiv.org/abs/1505.02074`.

[8] Z. Susskind, B. Arden, L. K. John, P. Stockton, and E. B. John. Neuro-symbolic ai: An emerging class of ai workloads and their characterization, 2021. URL `https://arxiv.org/abs/2109.06133`.

[9] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon. Cbam: Convolutional block attention module, 2018. URL `https://arxiv.org/abs/1807.06521`.

[10] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola. Stacked attention networks for image question answering, 2015. URL `https://arxiv.org/abs/1511.02274`.

[11] Z. Yang, J. Lee, and C. Park. Injecting logical constraints into neural networks via straight-through estimators. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 25096–25122. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/yang22h.html`.

[12] S. Zagoruyko and N. Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer, 2016. URL `https://arxiv.org/abs/1612.03928`.

[13] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization, 2015. URL `https://arxiv.org/abs/1512.04150`.