In Defense of Grid Features for Visual Question Answering

Huaizu Jiang[1,2]*, Ishan Misra[2], Marcus Rohrbach[2], Erik Learned-Miller[1], and Xinlei Chen[2]
[1]UMass Amherst, [2]Facebook AI Research (FAIR)
{hzjiang,elm}@cs.umass.edu, {imisra,mrf,xinleic}@fb.com

Abstract

*Popularized as 'bottom-up' attention [2], bounding box (or region) based visual features have recently surpassed vanilla grid-based convolutional features as the de facto standard for vision and language tasks like visual question answering (VQA). However, it is not clear whether the advantages of regions (*e.g.* better localization) are the key reasons for the success of bottom-up attention. In this paper, we revisit grid features for VQA, and find they can work surprisingly well – running more than an order of magnitude faster with the same accuracy (*e.g.* if pre-trained in a similar fashion). Through extensive experiments, we verify that this observation holds true across different VQA models (reporting a state-of-the-art accuracy on VQA 2.0 `test-std`, **72.71**), datasets, and generalizes well to other tasks like image captioning. As grid features make the model design and training process much simpler, this enables us to train them end-to-end and also use a more flexible network design. We learn VQA models end-to-end, from pixels directly to answers, and show that strong performance is achievable without using any region annotations in pre-training. We hope our findings help further improve the scientific understanding and the practical application of VQA. Code and features will be made available.*

1. Introduction

After the introduction of deep learning [9, 42] and attention mechanisms [45, 46] to multi-modal vision and language research, perhaps one of the most significant developments was the discovery of 'bottom-up' attention [2]. Unlike normal attention that uses 'top-down' linguistic inputs to focus on specific parts of the visual input, bottom-up attention uses pre-trained object detectors [31] to identify salient regions based *solely* on the visual input itself. As a result, images are represented by a collection of bounding box or **region**[1]-based features [2, 37]–in contrast to vanilla **grid** convolutional feature maps from ConvNets [33, 15]–for follow-up tasks. These region features have since then gained wide popularity and dominated vision and language leader boards [16, 48] for major tasks like visual question answering (VQA).

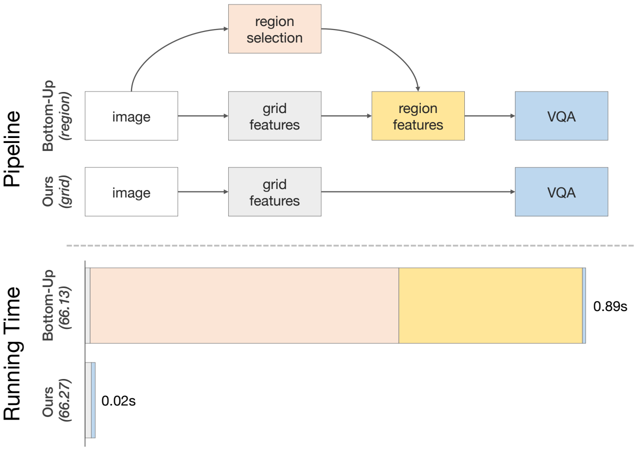

Figure 1: We revisit **grid**-based convolutional features for VQA, and find they can *match* the accuracy of the dominant **region**-based features from bottom-up attention [2], provided that one closely follow the pre-training process on Visual Genome [21]. As computing grid features skips the expensive region-related steps (shown in colors), it leads to significant speedups (all modules run on GPU; timed in the same environment).

So what makes these region features successful? Naturally, one would assume a major reason is better *localization* of individual objects, as the regions are direct bounding box outputs from detectors. Another plausible answer is that a number of regions can easily capture both the coarse-level information and fine-grained details in the image – even if they overlap. However, do these potential advantages actually demonstrate that region features are superior to grids?

Surprisingly, we discovered that grid features extracted from *exactly* the same layer of the pre-trained detector can perform competitively against their region-based counterparts for VQA. Moreover, with simple modifications during training, the same grid features can be made even more effective and that they consistently achieve comparable and sometimes better VQA accuracy than region features. In fact, our ablative analysis suggests that the key factors which contributed to the high accuracy of existing

*This work was done when Huaizu Jiang was an intern at FAIR.

[1] We use the terms 'region' and 'bounding box' interchangeably.

bottom-up attention features are: 1) the large-scale object and attribute annotations collected in the Visual Genome (VG) [21] dataset used for pre-training; and 2) the high spatial resolution of the input images used for computing features. As for the feature *format* itself – region or grid – it only affects accuracy *minimally*. Through a comprehensive set of experiments, we verified that our observations generalize across different network backbones, different VQA models [16, 48], different VQA benchmarks [3, 12], and even to other relevant tasks (*e.g*. image captioning [4]).

Our findings have important consequences for the design of future multi-modal vision and language models. The immediate benefit of switching to grids is inference speed, as we can now skip *all* of the region-related steps in the existing VQA pipeline (Fig. 1). For example, using a ResNet-50 [15] backbone, we find the overall running time drops from 0.89s to 0.02s per image – **40+** times faster with slightly better accuracy! In fact, extracting region features is so time-consuming that most state-of-the-art models [20, 48] are directly trained and evaluated on *cached* visual features. This practice not only imposes unnecessary constraints on model designs, but also limits potential applications of existing vision and language systems.

Empowered by grid features, we therefore take an initial step to train VQA models *end-to-end* from pixels directly to answers. Note that end-to-end training with region features is challenging, since fine-tuning region locations likely requires additional grounding annotations [13] that are computationally expensive and difficult to acquire. In contrast, grid features can be readily optimized for the final objective (*e.g*. to answer questions correctly) without extra grounding. The grid-feature pipeline also allows us to explore more effective designs for VQA (*e.g*. pyramid pooling module [52]) and enables networks pre-trained with **zero** region-level annotations to greatly reduce the gap in accuracy with VG models (trained on bounding boxes) – indicating strong VQA models can be achieved without *any* explicit notion of regions. These results further strengthen our defense of grid features for VQA. We hope our discovery can open up new opportunities for vision and language research in general.

2. Related Work

Visual features for vision and language tasks. Features have played a key role in the advancement of vision and language tasks. For example, deep learning features led to remarkable improvements in image captioning [9, 42, 8]. While a complete review of visual features used for vision and language tasks is beyond the scope of this paper, we note that the accuracies of modern VQA models are dependent on the underlying visual features used, including VGG [33] and ResNet [15] grid features, which were later dominated by bottom-up attention region features [2, 37]. Today, most state-of-the-art VQA models focus on fusing schemes [49, 20, 48] and are built with region features as-is [47]; whereas our work revisits grid features, and shows that they can be equally effective and lead to remarkable speed-ups – often greater than an order of magnitude!

Pre-training for VQA. Most VQA methods use two separately pre-trained models: vision models trained on ImageNet [6] and VG [21]; and word embeddings [29] for linguistic features. As these separately trained features may not be optimal for joint vision and language understanding, a recent hot topic is to develop jointly pre-trained models [23, 27, 36, 35, 53, 5] for vision and language tasks. A common scheme for such methods is to view regions and words as 'tokens' for their respective domain, and pre-train a variant of BERT [7, 40] for 'masked' token prediction. Complementary to that direction, our work delves specifically into the 'format' of visual tokens and can be potentially combined with such methods for mutual benefits (*e.g*. trade-off between speed and accuracy).

Regions *vs*. grids. The debate between region features and grid features carries some inherent connections to object detection: the dominance of the R-CNN based detection models [31, 14] demonstrates that a region (the 'R' in R-CNN) based refinement stage is beneficial for object detection. On the other hand, one-stage detectors [24, 26] approach the detection task *without* the need for explicit region-level computation and show that grid features can be competitive for object detection. In our work, we also use grid features – *no* regions for the VQA task. To minimize changes from bottom-up attention paper [2], we pre-train the features with Faster R-CNN [31]. However, during inference, we discard the region-related steps from the detector and use *only* the grid convolutional features. This in fact gives us a *stronger* defense for grids, as we show that VQA can operate on a 'single' feature map, instead of feature maps of 'multiple' scales that one-stage detectors [24, 26] thrive on.

It is also worth noting that while region features are effective on benchmarks like VQA [3, 11] and COCO captions [4], for benchmarks that diagnose a model's reasoning abilities when answering visual questions (*e.g*. CLEVR [17]), simple methods based on grids [30] have shown strong performance. We hope that our discovery that grid features also work well for the general VQA task can bridge the gap between these two lines of work [32].

3. From Regions to Grids

In this section, we explain our approach to obtaining grid features that are just as effective as region features, with the constraint that they have been pre-trained with the *same* task. In Sec. 7, we show that the 'same pre-training' constraint can be lifted and grid features can still close the gap to regions with end-to-end training on down-stream tasks. We first briefly review the region features from bottom-up attention [2].

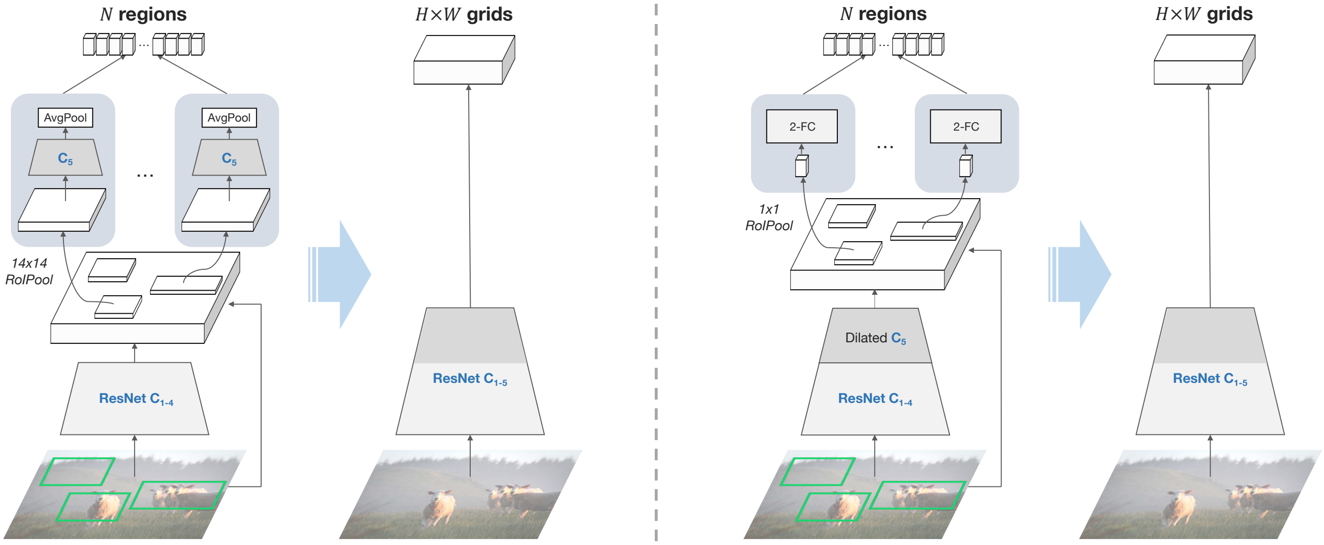

Figure 2: From regions to grids. Left: We convert the original region feature extractor used by bottom-up attention [2] back to the ResNet [15] grid feature extractor for the *same* layer (see Sec. 3.2, weights in blue are transferred), and find it works surprisingly well for VQA [11]. **Right**: We build a detector based on 1×1 `RoIPool` while keeping the output architecture *fixed* for grid features (see Sec. 3.3), and the resulting grid features consistently perform at-par with region features.

3.1. Bottom-Up Attention with Regions

The bottom-up attention method [2] uses a Faster R-CNN [31] detection model. The detector is trained on a cleaned version of Visual Genome [21], with thousands of object categories and hundreds of attributes with bounding box (region) annotations.

In order to obtain bottom-up attention features for tasks like VQA, two region-related steps are needed:

Region selection. As Faster R-CNN is a two-stage detector, region selection happens twice in the pipeline. The first is through a region proposal network [31], which deforms and selects prominent candidate 'anchors' as Regions of Interest (RoIs). Another selection is done as post-processing to aggregate top N boxes in a *per-class* manner. In both steps, non-maximal suppression (NMS) is used, which keeps the region with the highest classification score and removes other near-duplicates in a local neighborhood.

Region feature computation. Given regions from the first stage (up to thousands), `RoIPool` operations [31] are used to extract the initial region-level features. Additional network layers then compute the output representation of regions *separately*. Finally, region features that survive both rounds of selection are stacked together as the bottom-up features to represent an image.

It is important to note that due to the complexity of the VG dataset (*e.g*. thousands of classes) and the specific Faster R-CNN detector used [2] (described next), both steps are computationally intensive. In contrast, directly using grid features can skip or accelerate these steps and offer potentially significant speed-ups.

3.2. Grid Features from the Same Layer

The simplest way to convert region features to grids is to see if one can directly compute outputs of the same network layer, but in a *shared*, fully convolutional manner. To this end, we take a closer look at the specific Faster R-CNN architecture used by the original bottom-up attention [2].

The Faster R-CNN is a variant of the *c4* model [15] with an extra branch for attribute classification. It divides the weights from a ResNet [15] into two separate sets: given an input image, it first computes feature maps using the lower blocks of ResNet up to C_4. This feature map is shared among all regions. Then, separately, per-region feature computations are performed by applying the C_5 block on the 14×14 `RoIPool`-ed features. The output of C_5 is then *AvgPool*-ed to a final vector for each region as the bottom-up features [2]. Since all the final region features are from C_5, it is easy to convert the detector *back* to the ResNet classifier and take the same C_5 layer as our output grid features. Fig. 2 (left) illustrates our conversion process.

As our experiments will show, directly using the converted C_5 output already works surprisingly well. Any performance drop from doing so may be because Faster R-CNN is highly optimized for *region*-based object detection, and likely not so much for grids. Therefore, we next see if some minimal adjustments to the model can be made to improve grid features.

3.3. 1×1 `RoIPool` for Improved Grid Features

Our idea is to simply use 1×1 `RoIPool`. This means representing each region with a single *vector*, rather than a

		VG detection pre-train			VQA	
#	feature	RoIPool	region layers	AP	accuracy	Δ
1	R [2]	14×14	C_5 [15]	4.07	64.29	-
2		1×1	2-FC	2.90	63.94	*-0.35*
3	G	14×14	C_5	4.07	63.64	*-0.65*
4		1×1	2-FC	2.90	**64.37**	*0.08*
5		ImageNet pre-train			60.76	*-3.53*

Table 1: Main comparison. 'R' stands for region features as in bottom-up attention [2]. 'G' stands for grid features. All results reported on VQA 2.0 vqa-eval. We show that: **1)** by simply extracting grid features from the *same* layer C_5 of the same model, the VQA accuracy is already much closer to bottom-up attention than ImageNet pre-trained ones (row 1,3 & 5); **2)** 1×1 RoIPool based detector pre-training improves the grid features accuracy while the region features get worse (row 1,2 & 4). Last column is the gap compared to the original bottom-up features (underlined).

three-dimensional tensor in Faster R-CNN. At first glance, it may seem counter-intuitive, as the two additional spatial dimensions (height and width) are useful to characterize different parts of objects in 2D – indeed, we find this modification negatively affects object detection performance on VG. But importantly, using 1×1 RoIPool regions also means each vector on the grid feature map is *forced* to cover all the information for a spatial region *alone*, which can potentially result in stronger grid features.

However, directly applying 1×1 RoIPool on the original model is problematic, likely because C_5 consists of several ImageNet pre-trained convolutional layers that work best with inputs of particular spatial dimensions. To resolve this, we follow recent developments in object detection and use the entire ResNet up to C_5 as the backbone for shared feature computation [54]; and for region-level computation place two 1024D fully-connected (FC) layers on the top, which by *default* accept vectors as inputs.

To reduce the effect of low resolutions when training the detector with features pooled from C_5 (C_5 has stride 32, whereas C_4 has 16), the stride-2 layers are replaced with stride-1 layers, and the remaining layers are dilated with a factor of 2 [54]. For grid feature extraction, we remove this dilation and convert it back to the normal ResNet.

Fig. 2 (right) summarizes the changes we made to improved grids. Note that compared to the original model (left), we only made necessary modifications to the region related components during training. Since all such computations are removed during feature extraction, our grid feature extractor is kept *untouched* during inference.

4. Main Comparison: Regions *vs*. Grids

From this section on, we report our experimental results comparing regions with grids. We choose VQA (2.0) [11] as our main task of interest, since it is currently a major benchmark for evaluating joint vision and language understanding and has clear metrics for evaluation. For all our comparisons, we denote methods using region features with the tag 'R', and methods using grid features with 'G'. In this section, we focus on reporting our main findings from converting regions to grids as described in Sec. 3. We begin by briefly describing our experimental setups (more details in the supplementary material). Note that our goal here is to make the conclusion *meaningful* by controlled comparisons, and not necessarily to optimize for absolute performance.

4.1. Experimental Setup

Faster R-CNN. For analysis, we use Faster R-CNN with a ResNet-50 backbone pre-trained on ImageNet by default[2]. Closely following bottom-up attention [2], the detector is then trained on the VG dataset [21] with region-level annotations for 1600 object categories and 400 attribute classes. For attributes, an additional branch is added with loss weight 0.5. The model is trained with '1x' schedule [14]. Notably, input images are resized to have a maximum shorter side of 600 pixels (longest 1000) when keeping aspect ratio fixed. For region features, we set N=100.

VQA split. Unless otherwise specified, we use the default train set for training. To assist our analysis, we create a local validation set, vqa-dev, out of the standard val set to select the best model during training for evaluation. It contains randomly sampled 8.4K images and their corresponding questions, with 66K pairs in total. The rest of the original val set (named vqa-eval) is reserved for testing, on which we report results.

VQA model. We use the co-attention model [50] implemented in Pythia [16, 34]. This model fuses visual features (either region or grid) with textual representations of questions, and outputs the final answer.

4.2. Main Results

Our main results are summarized in Table 1. We make two observations: First, compared with the widely used bottom-up region features (row 1), directly extracting outputs from C_5 with the same model (row 3) works *surprisingly* well (64.29 *vs*. 63.64 accuracy). In contrast, the standard ResNet-50 model pre-trained on ImageNet [6] shows much worse performance – 60.76 accuracy, a gap of more than 3% with the bottom-up features.

Second, while our 1×1 RoIPool-based variant hurts the object detection performance (average precision [25] on VG drops from 4.07 to 2.90), it helps VQA – boosting the accuracy by 0.73% (row 3 & 4) and as a result slightly *outperforms* the original region-based features. On the other hand, our RoI-based variant does not help the region features method and drops the accuracy of region features to 63.94. This indicates the original model used by bottom-up attention favors regions; while our design works better for grids. Thus, we use the setting of the 1^{st} row (best for

[2] https://github.com/facebookresearch/maskrcnn-benchmark

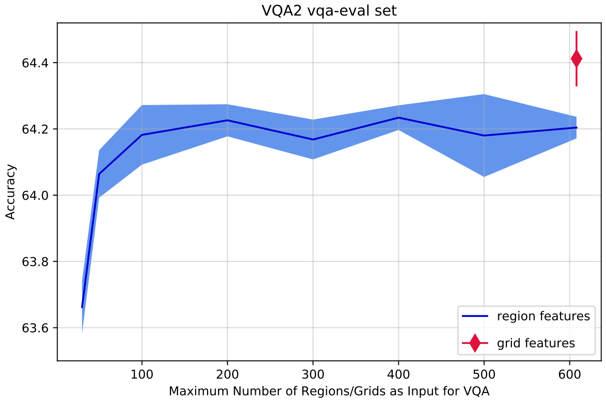

Figure 3: VQA accuracy *vs.* number of features N as input to the VQA model. We report the average accuracy and standard deviation across 5 independent runs on the VQA 2.0 `vqa-eval` set. We observe that the VQA accuracy of region features saturates around 200 regions. In contrast, the grid features benefit from a larger N (translates from a larger input size) and in this case stays better than regions even when N is the same (608).

regions) to represent 'R', and the 4^{th} row (best for grids) to represent 'G', to perform a more in-depth study and fair comparison between the two through the rest of the paper.

4.3. Number of Regions

Apart from architectural differences in training, another factor that can affect VQA accuracy is the number of feature vectors N used to represent images. Our region model from Pythia [16] has a default setting that uses the top 100 boxes to represent region features, increasing it from the original 36 boxes in [2] to improve the accuracy. On the other hand, since grid features are convolutional feature maps for a preset layer, the number of features is determined by the input size to the network. As our largest input size is 600×1000, a 32-stride feature map (C_5) results in 608 grid features – much larger than the number of region features. To understand how these different numbers of region features affect the accuracy, we ran experiments with varying number of features N and show the results in Figure 3.

As for the region features, we observe an improvement in accuracy as the number of regions increases from 30 to 200, beyond which the accuracy saturates. Interestingly, our grid features are better even when compared to the highest number of regions[3]. Thus, the higher number of feature vectors used in our grid method compared to the baseline region method, is not the reason for its improved VQA accuracy.

4.4. Test Accuracy and Inference Time

We now report results on the VQA 2.0 `test-dev` set to quantify the difference in performance between region and grid features. Note that different from previous setups, we use `trainval+vqa-eval` for training. We report the VQA accuracy and the inference time breakdown in Table 2. Unlike our grid features which directly use convolutional feature maps, region features involve additional operations of region selection and region feature computation. These additional operations take 98.3% of the total inference time for a region-based model. As a result, the VQA model that takes our grid features as input runs **48**× faster than its counterpart using bottom-up region features.

	# features (N)	`test-dev` accuracy	inference time breakdown (ms)				
			shared conv.	region feat. comp.	region selection	VQA	total
R	100	66.13	9	326	548	6	889
	608	66.22	9	322	544	7	882
G	608	66.27	11	-	-	7	18

Table 2: Region *vs.* grid features on the VQA 2.0 `test-dev` with accuracy and inference time breakdown measured in milliseconds per image. Our grid features achieve comparable VQA accuracy to region features while being much faster without region feature computation and region selection.

4.5. Qualitative Comparison

We visualize attention maps over input images from the top-down attention module [2], together with answers from both regions and grids in Fig. 4. Source images are taken from COCO [25] on which VQA 2.0 [11] benchmark is built. To obtain the attention map, we propagate the attention value of each region or grid to its corresponding pixels, and then average the attention value for each pixel (normalizing them individually to [0, 1]). As can be seen, both types of features are able to capture relevant concepts in input images (*e.g.*, snowfield in the top left). Naturally, attention maps of region features tend to cover object-like regions, while for grid features the attention does not necessarily cover the full area the supporting concept (*e.g.*, the snowfield), which can be used to answer the question. However, both features are able to answer visual questions well, suggesting that localization is important, but accurate object detection of individual objects is not crucial for VQA [11].

We show failure cases of region and grid features in Fig. 4 (b)(c)(d). In most examples, the models attend to the supporting concepts but still give wrong answers. In the cases where both region and grid features fail, specifically designed modules may be needed (*e.g.*, counting module [51, 39] in the bottom right example) to answer the question correctly.

5. Why do Our Grid Features Work?

As we mentioned in Sec. 2, grid features are not new – in fact, they were widely used in vision and language tasks before the introduction of bottom-up attention features. Com-

[3] Since NMS is used in selecting regions, the maximum number N varies across images. Therefore we 1) cannot directly set it to the same number as grids and 2) report maximum N instead (zero paddings are used for images with fewer regions).

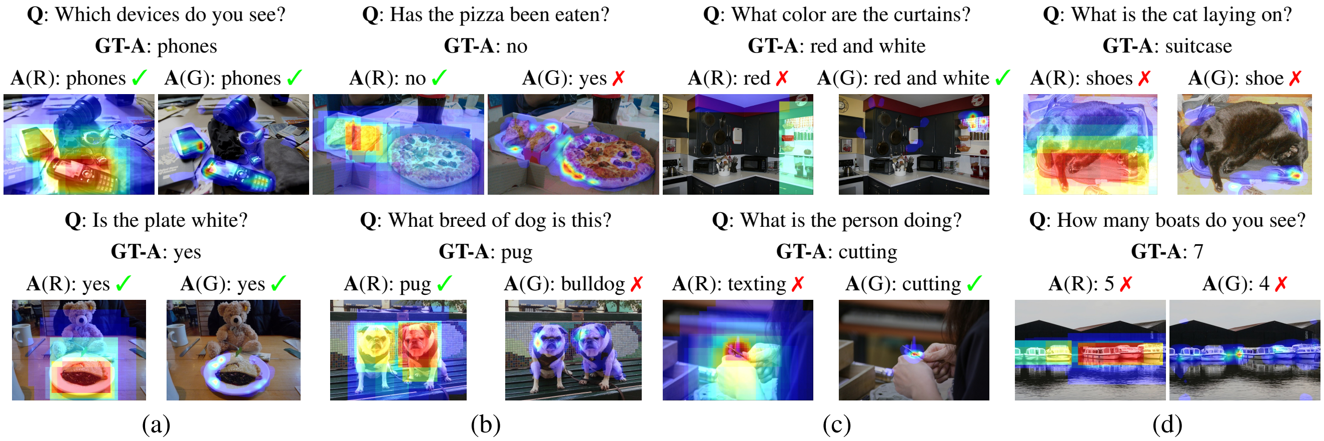

Figure 4: Visualizations of attention maps overlaid on images produced by VQA models [16]. Source images taken from COCO [25] to compare against bottom-up attention [2] on VQA 2.0 [11]. We show questions (Q), ground-truth answers (GT-A), and side-by-side predictions (attention maps, answers) of region (R) and grid (G) features. **From left to right**: (a) both region and grid features give correct answers, (b) region features give correct answers but grid features fail, (c) region features fail but grid features give correct answers, and (d) both region and grid features fail. Best viewed in color.

		accuracy	pre-training task	input size
G	*prev.*	60.76	ImageNet [6] classification	448×448
	ours	64.37	VG [21] object+attribute detection	600×1000

Table 3: Comparison between the conventional **ImageNet pre-trained and our proposed grid features** on the VQA 2.0 `vqa-eval` set. Besides VQA accuracy, we list two major differences between the two: 1) pre-training task and 2) input image size.

pared to the previous attempts at grid features, why do *our* grid features work well? In Table 3 we show the performance of grid-based methods (ResNet-50 C_5 features) for different settings and find that there are two major factors: 1) input image size; 2) pre-training task. We study both these factors next and report results on the `vqa-eval` set.

5.1. Factor 1: Input Image Size

The standard image size used during feature extraction for ImageNet pre-trained models is 448×448 [10] discarding the aspect ratio; whereas for VG detection in bottom-up attention [2], the default size is 600×1000 while keeping the aspect ratio intact. Therefore, we experimented with different combinations and reported results for all of them in Table 4. We note that for grid features, a larger input size means more features for the VQA model.

From the table, we find that grid features benefit from larger images as input, indicating this factor is indeed important. However, input size has a different effect for models pre-trained on ImageNet *vs*. VG. For ImageNet models which are pre-trained on smaller images [15], the performance saturates around 600×1000. Interestingly, the performance of VG models improves with the input size and continues to increase even at 800×1333. We still use 600×1000 for the rest of the paper.

	dataset	input size		# features	accuracy
		shorter side	longer side	N	
G	ImageNet	448	448	196	60.76
		448	746	336	61.21
		600	1000	608	61.52
		800	1333	1050	61.52
	VG	448	448	196	63.24
		448	746	336	63.81
		600	1000	608	64.37
		800	1333	1050	64.61

Table 4: Impact of input image size on the VQA 2.0 `vqa-eval` set. Grid features benefit from larger input image sizes. For an ImageNet pre-trained model, the accuracy saturates around 600×1000 but the VG model makes a better use of larger input image sizes.

5.2. Factor 2: Pre-Training Task

We now study the difference in VQA accuracy due to the pre-training task in the ImageNet (classification) and VG (detection)[4]. To understand these differences better, we introduce an additional pre-trained model in each setting. For classification, we include a model trained on YFCC [38], which has 92M images with image tags. For detection, we include a standard model from COCO [25] which only has object annotations (no attributes). All models use a ResNet-50 backbone for fair comparison.

The results are shown in Table 5. In the image classification pre-trained setting, the YFCC model (trained on weak image level tags), performs better than the ImageNet model, possibly because it is trained on two orders of magnitude more data. For detection based pre-training, the VG model (trained with objects and attributes) gives better results than

[4] Strictly speaking, VG also uses ImageNet classification for pre-training, because the detector is fine-tuned from a standard ImageNet pre-trained model.

	pre-train task				accuracy
	setup	dataset	annotation	#images	
G	cls	ImageNet [6]	image label	1.3M	61.52
	cls	YFCC [38]	image tag	92M	62.72
	det	COCO [25]	object box	118K	62.46
	det	VG [21]	object+attribute	103K	64.37

Table 5: Choice of pre-training task. We explore the impact of the type of pre-training task on the final performance while keeping the input size fixed at 600×1000. Results reported on `vqa-eval`. We broadly characterize the pre-training tasks into two types - object detection ('det') and image classification ('cls').

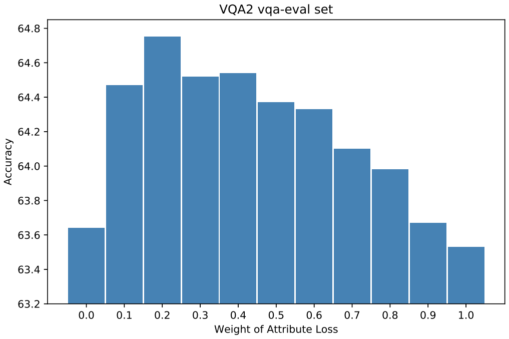

Figure 5: Analysis on **attribute loss weights** when pre-training grid features on Visual Genome (VG). All results on VQA 2.0 `vqa-eval` set.

the COCO model. The larger number of categories in VG compared to COCO (1600 *vs*. 80) or the additional attribute annotations it has are two possible reasons for the improved performance. We study the impact of attributes next.

Attributes. Fig. 5 shows the impact of the attribute loss weight on VQA accuracy. Setting the attribute loss weight to zero during pre-training on VG, results in a drop in VQA performance. In fact, the VQA accuracy in this case matches the accuracy from a pre-trained COCO model suggesting that attributes in the pre-training task are a major reason for the better performance of VG models. We also note that the grid features consistently outperform the region features for all values of the attribute loss weight.

6. Generalization of Grid Features

We now study whether our findings about grid features are more broadly applicable to other tasks and models. In this section, we study generalization across: 1) different backbones; 2) different VQA models; 3) different VQA tasks; 4) other tasks. For all the studies, we set the attribute loss weight to 0.2, and compare both the accuracy and speed. For regions we use top N=100 ones. Detailed hyper-parameters are in the supplementary material.

Different backbone. We train Faster R-CNN models with ResNeXt-101-32x8d [44] backbone on VG and use the same Pythia setting from Section 4.5. Results on VQA 2.0 `test-dev` split are reported in Table 6a. We find that our grid features are competitive to the region features even on this more powerful backbone model. Speed-wise, grid features still run substantially faster (23.8×) than region ones.

Different VQA model. We further test our features obtained from the previous ResNeXt-101 backbone with the state-of-the-art VQA model, MCAN [48] (2019 VQA Challenge winner). We use the open-sourced implementation[5] to train the *large* version of the model. The results on VQA 2.0 `test-dev` set are in Table 6b, where our own region features perform better than the results reported in [48] due to stronger backbone. On top of that, our grid features work even *better* than regions, leading to significant improvement over results reported in MCAN [48] (+1.66). This final model reports a state-of-the-art `test-std` result of **72.71** (single-model performance) for future reference.

Different VQA task. We use the VizWiz VQA dataset [12], which is a real world dataset of pictures taken with cellphones by visually-impaired users. It is more challenging due to poor image quality, conversation-style questions, and unanswerable questions, *etc*. Pythia [16] model is used (2018 challenge winner). Results on the `test-dev` set of VizWiz are reported in Table 6c, where our grid features achieve comparable results to the regions. It is worth pointing out that our grid features run much faster (23×), which provides great potential to be deployed in practice, *e.g*., on cell phones, to better assist the visually-impaired.

Image captioning. We train the bottom-up attention model [2] implemented in Pythia [16] taking our features as input for image captioning on COCO [4]. No CIDEr [41] optimization [2] is used for fair comparison. Quantitative results on the test set of Karpathy split [18] are reported in Table 6d. We use standard evaluation metrics including BLEU4 [28], METEOR [22], CIDEr, and SPICE [1]. Similar to the VQA task, our grid features achieve comparable results to bottom-up region ones for image captioning while being significantly faster.

7. Towards End-to-end VQA

Although pre-training on VG, ImageNet, or YFCC provides useful feature representations for VQA, there are still potential domain shifts between the pre-training tasks and the downstream tasks. For example, YFCC contains a lot of outdoor images [38], which are not present in the VQA dataset. Instead of using pre-computed fixed feature representations, *end-to-end* training, where the initial feature representations will be fine-tuned, provides a natural solution to reducing such domain gaps. Empowered by the dramatic simplification of grid features for the VQA pipeline, we take an initial step towards this goal.

[5] https://github.com/MILVLG/mcan-vqa

	accuracy	time (ms)
Pythia [16]	68.31	-
R	68.21	929
G	67.76	39

(a)

	accuracy	time (ms)
MCAN [48]	70.93	-
R	72.01	963
G	72.59	72

(b)

	accuracy	time (ms)
Pythia [16]	54.22	-
R	54.28	874
G	54.17	38

(c)

	B4	M	C	S	time (ms)
BUTD [2]	36.2	27.0	113.5	20.3	-
R	36.2	27.7	113.9	20.8	1101
G	36.4	27.4	113.8	20.7	240

(d)

Table 6: Generalizations of grid features. From left to right: (a) Different **backbone.** We use a ResNeXt-101-32x8d instead of a ResNet-50 as the backbone. (b) Different **VQA model**. We use MCAN [48] implementation which is the state-of-the-art VQA model. (c) Accuracy on **VizWiz** using the same VQA models [16]. (d) **Image captioning** on COCO Karpathy test split. Abbreviations: BLEU4 (B4), METEOR (M), CIDEr (C), and SPICE (S). Our grid features generalize well by achieving results at-par with bottom-up region features while being significantly faster.

pre-train task		e2e	PPM [52]	accuracy	Δ
dataset	region annotations?				
VG [21]	✓			66.27	-
		✓		66.47	*0.20*
		✓	✓	66.74	*0.47*
ImageNet [6]	✗			63.21	-
		✓		64.98	*1.77*
		✓	✓	65.97	*2.76*
YFCC [38]	✗			65.04	-
		✓		65.35	*0.31*
		✓	✓	66.61	*1.57*

Table 7: Results of **end-to-end trained VQA** models with grid features on the VQA 2.0 `test-dev` set. End-to-end learning boosts accuracy for all models and more for ones trained on ImageNet and YFCC. Adding PPM [52] further improves accuracy.

Training details. We adopt the 22K learning rate schedule [16] to train both the ResNet-50 model and the Pythia VQA model jointly, with errors from the answering accuracy *directly* back-propagated to the grid convolutional feature maps. We fix the first two residual blocks and fine-tune the rest of the model. Since the visual representations are computed online (not stored on disk), it allows us to perform data augmentation including color jitter and affine transformation over the input images to reduce chance of over-fitting. For more details see supplementary material.

Results. We experiment with three models pre-trained on VG, ImageNet, and YFCC. Note that while VG uses *region*-level annotations, both ImageNet and YFCC only use *image*-level ones (human labels or noisy image tags). As can be seen from Table 7, end-to-end training (denoted as 'e2e') can boost accuracy for all three pre-trained models, with the biggest improvements for ImageNet models.

Flexible network design. As we now have the ability to train our models end-to-end in a simple manner, it allows us to introduce more flexible architectural designs for vision and language tasks [27]. Specifically, on top of the grid features from the ResNet-50 model, we add a Pyramid Pooling Module (PPM, a component widely used for semantic segmentation; details in supplementary material) [52, 43] to aggregate visual information from grid features of different spatial resolutions. After adding this module to different pre-trained models (Table 7, 'PPM'), the VQA accuracy can be further improved. Remarkably, for ImageNet and YFCC pre-trained models, a combination of end-to-end training and PPM results in close or even *better* performance than a VG pre-trained model using pre-computed region features. This result is particularly desirable as it indicates good VQA accuracy can be achieved even with *zero* use of explicit region (bounding box) annotations.

8. Conclusion

In this paper, we revisit grid features as an alternative to the widely used bottom-up region features [2] for vision and language tasks. We show they can in fact achieve on-par results in terms of accuracy over different VQA tasks and models and even on captioning. As a result of skipping the computationally expensive region-related bottlenecks in the pipeline, we see remarkable speed-ups – often more than an order of magnitude – to the existing systems that rely on regions. Our experiments show that rather than the 'format' of features (region *vs*. grids), the semantic content that features represent is more critical for their effectiveness. Such effective representation, per our experiment, can be achieved either by pre-training on object and attribute datasets such as VG, or more importantly, by *end-to-end* training of grid features directly for the end-task. Note that while easy with grid-features, end-to-end training is not trivial with regions. Even with limited exploration in this direction, we already find that given a more flexible design space, grid features pre-trained without *any* region-level annotations can in fact achieve strong performance on VQA. While we are aware that for tasks like referring expressions [19] where the output itself is a region, modeling region is likely unavoidable, but we hope our grid features can potentially offer new perspectives for vision and language research in general.

Acknowledgements. We would like to thank Larry Zitnick, Devi Parikh, and Amanpreet Singh for helpful discussions. ELM acknowledges support from AFRL and DARPA (#FA8750- 18-2-0126). The U.S. Gov. is authorized to reproduce and distribute reprints for Gov. purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the AFRL and DARPA or the U.S. government.

References

[1] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. In *ECCV*, 2016. 7

[2] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*, 2018. 1, 2, 3, 4, 5, 6, 7, 8

[3] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *ICCV*, 2015. 2

[4] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015. 2, 7

[5] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Learning universal image-text representations. *arXiv preprint arXiv:1909.11740*, 2019. 2

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2, 4, 6, 7, 8

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2018. 2

[8] Jacob Devlin, Saurabh Gupta, Ross Girshick, Margaret Mitchell, and C Lawrence Zitnick. Exploring nearest neighbor approaches for image captioning. *arXiv preprint arXiv:1505.04467*, 2015. 2

[9] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 1, 2

[10] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*, 2016. 6

[11] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *CVPR*, 2017. 2, 3, 4, 5, 6

[12] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *CVPR*, 2018. 2, 7

[13] Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 1990. 2

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 2, 4

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 3, 4, 6

[16] Yu Jiang, Vivek Natarajan, Xinlei Chen, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. Pythia v0.1: the winning entry to the vqa challenge 2018. *arXiv preprint arXiv:1807.09956*, 2018. 1, 2, 4, 5, 6, 7, 8

[17] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017. 2

[18] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015. 7

[19] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014. 8

[20] Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. Bilinear attention networks. In *NeurIPS*, 2018. 2

[21] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 2017. 1, 2, 3, 4, 6, 7, 8

[22] Alon Lavie and Abhaya Agarwal. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *ACL*, 2007. 7

[23] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019. 2

[24] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 2

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 4, 5, 6, 7

[26] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 2

[27] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*, 2019. 2, 8

[28] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002. 7

[29] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014. 2

[30] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018. 2

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 1, 2, 3

[32] Robik Shrestha, Kushal Kafle, and Christopher Kanan. Answer them all! toward universal visual question answering models. In *CVPR*, 2019. 2

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 2

[34] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *CVPR*, 2019. 4

[35] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019. 2

[36] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019. 2

[37] Damien Teney, Peter Anderson, Xiaodong He, and Anton van den Hengel. Tips and tricks for visual question answering: Learnings from the 2017 challenge. In *CVPR*, 2018. 1, 2

[38] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016. 6, 7, 8

[39] Alexander Trott, Caiming Xiong, and Richard Socher. Interpretable counting for visual question answering. In *ICLR*, 2018. 5

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2

[41] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *CVPR*, 2015. 7

[42] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015. 1, 2

[43] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018. 8

[44] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 7

[45] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015. 1

[46] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *CVPR*, 2016. 1

[47] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic VQA: disentangling reasoning from vision and language understanding. In *NeurIPS*, 2018. 2

[48] Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. Deep modular co-attention networks for visual question answering. In *CVPR*, 2019. 1, 2, 7, 8

[49] Zhou Yu, Jun Yu, Jianping Fan, and Dacheng Tao. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In *ICCV*, 2017. 2

[50] Zhou Yu, Jun Yu, Chenchao Xiang, Jianping Fan, and Dacheng Tao. Beyond bilinear: Generalized multimodal factorized high-order pooling for visual question answering. *TNNLS*, 2018. 4

[51] Yan Zhang, Jonathon S. Hare, and Adam Prügel-Bennett. Learning to count objects in natural images for visual question answering. In *ICLR*, 2018. 5

[52] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 2, 8

[53] Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason J Corso, and Jianfeng Gao. Unified vision-language pre-training for image captioning and vqa. *arXiv preprint arXiv:1909.11059*, 2019. 2

[54] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019. 4