INTERPRETABLE COUNTING IN VISUAL QUESTION ANSWERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Questions that require counting a variety of objects in images remain a major challenge in visual question answering (VQA). The most common approaches to VQA involve either classifying answers based on fixed length representations of both the image and question or summing fractional counts estimated from each section of the image. In contrast, we treat counting as a sequential decision process and force our model to make discrete choices of what to count. Specifically, the model sequentially selects from detected objects and uses inferred relationships between objects to influence subsequent selections. A distinction of our approach is its intuitive and interpretable output, as discrete counts are automatically grounded in the image. Furthermore, our method outperforms the state of the art architecture for VQA on multiple metrics that evaluate counting.

1 INTRODUCTION

Visual question answering (VQA) is an important benchmark to test for context-specific reasoning over complex images. While the field has seen substantial progress, counting-based questions have seen the least improvement (Chattopadhyay et al., 2017). Intuitively, counting should involve finding the number of distinct scene elements or objects that meet some criteria, see Fig. 1 for an example. In contrast, the predominant approach to VQA involves representing the visual input with the final feature map of a convolutional neural network (CNN), attending to regions based on an encoding of the question, and classifying the answer from the attention-weighted image features (Xu & Saenko, 2015; Yang et al., 2015; Xiong et al., 2016; Lu et al., 2016b; Fukui et al., 2016; Kim et al., 2017). Our intuition about counting seems at odds with the effects of attention, where a weighted average obscures any notion of distinct elements. As such, we are motivated to re-think the typical approach to counting in VQA and propose a method that embraces the discrete nature of the task.

Our approach is partly inspired by recent work that represents images as a set of distinct objects, as identified by object detection (Anderson et al., 2017), and making use of the relationships between these objects (Teney et al., 2016). We experiment with counting systems that build off of a vision module designed for object detection and relationship extraction, which we adapt from the recent Region-based Fully Convolutional Network (Dai et al., 2016).

For training and evaluation, we create a new dataset, HowMany-QA. It is taken from the counting-specific union of VQA 2.0 (Goyal et al., 2017) and Visual Genome QA (Krishna et al., 2016).

We introduce the Interpretable Reinforcement Learning Counter (IRLC), which treats counting as a sequential decision process. We treat learning to count as learning to enumerate the relevant objects in the scene. As a result, IRLC not only returns a count but also the objects supporting its answer. This output is produced through an iterative method. Each step of this sequence has two stages: First, an object is selected to be added to the count. Second, the model adjusts the priority given to unselected objects based on their inferred relationships to the selected objects (Fig. 1). We supervise only the final count and train the decision process using reinforcement learning (RL).

Additional experiments highlight the importance of the iterative approach when using this manner of weak supervision. Furthermore, we train the current state of the art model for VQA on HowMany-QA and find that IRLC achieves a higher accuracy and lower count error. Lastly, we compare the grounded counts of our model to the attentional focus of the state of the art baseline to demonstrate the interpretability gained through our approach.

Figure 1: IRLC takes as input a counting question and image. Detected objects are added to the returned count through a sequential decision process. The above example illustrates actual model behavior after training.

2 Related Work

Visual representations for counting. As a standalone problem, counting from images has received some attention but typically within specific problem domains. Segui et al. (2015) explore training a CNN to count directly from synthetic data. Counts can also be estimated by learning to produce density maps for some category of interest (typically people), as in Lempitsky & Zisserman (2010); Oñoro-Rubio & López-Sastre (2016); Zhang et al. (2015). Density estimation simplifies the more challenging approach of counting by instance-by-instance detection (Ren & Zemel, 2017). Methods to detect objects and their bounding boxes have advanced considerably (Girshick et al., 2015; Girshick, 2015; Ren et al., 2015b; Dai et al., 2016; Lin et al., 2017) but tuning redundancy reduction steps in order to count is unreliable (Chattopadhyay et al., 2017). Here, we overcome this limitation by allowing flexible, question specific interactions during counting.

Alternative approaches attempt to model subitizing, which describes the human ability to quickly and accurately gauge numerosity when at most a few objects are present. Zhang et al. (2017) demonstrates that CNNs may be trained towards a similar ability when estimating the number of salient objects in a scene. This approach was extended to counting nearly 100 classes of objects simultaneously in Chattopadhyay et al. (2017). Their model is trained to estimate counts within each subdivision of the full image, where local counts are typically within the subitizing range. In addition, Chattopadhyay et al. (2017) examine their model in the context of VQA counting questions and demonstrate impressive performance. However, their analysis was limited to the specific subset of examples where their approach was applicable.

Visual question answering. We are interested in the challenge created by incorporating question answering. The potential of deep learning to fuse visual and linguistic reasoning has been recognized for some time (Socher et al., 2014; Lu et al., 2016a). Visual question answering poses the challenge of retrieving question-specific information from an associated image, often requiring complex scene understanding and flexible reasoning. In recent years, a number of datasets have been introduced for studying this problem (Malinowski & Fritz, 2014; Ren et al., 2015a; Zhu et al., 2015; Agrawal et al., 2015; Goyal et al., 2017; Krishna et al., 2016). The majority of recent progress has been aimed at the so-named "VQA" datasets (Agrawal et al., 2015; Goyal et al., 2017), where counting questions represent roughly 11% of the data. Though our focus is on counting questions specifically, prior work on VQA is highly relevant.

An early baseline for VQA represents the question and image at a coarse granularity, respectively using a "bag of words" embedding along with spatially-pooled CNN outputs to classify the answer (Zhou et al., 2015). In Ren et al. (2015a), a similar fixed-length image representation is fused with the question embeddings as input to a recurrent neural network (RNN), from which the answer is classified.

Attention. More recent variants have chosen to represent the image at a finer granularity by omitting the spatial pooling of the CNN feature map and instead use attention to focus relevant image regions before producing an answer (Xu & Saenko, 2015; Yang et al., 2015; Xiong et al., 2016; Lu et al., 2016b; Fukui et al., 2016; Kim et al., 2017). These works use the spatially-tiled feature vectors output by a CNN to represent the image; others follow the intuition that a more meaningful representation may come from parsing the feature map according to the locations of objects in the scene (Shih et al., 2015; Ilievski et al., 2016). Notably, using object detection was a key design choice for the winning submission for the VQA 2017 challenge (Anderson et al., 2017). Work directed at VQA with synthetic images (which sidesteps the challenges created by computer vision) has further demonstrated the utility that relationships may provide as an additional form of image annotation (Teney et al., 2016).

Interpretable VQA. The use of "scene graphs" in real-image VQA would have the desirable property that intermediate model variables would be grounded in concepts explicitly, a step towards making neural reasoning more transparent. A conceptual parallel to this is found in Neural Module Networks (Andreas et al., 2016a;b; Hu et al., 2017), which gain interpretability by grounding the reasoning process itself in defined concepts. The general concept of interpretable VQA has been the subject of recent interest. Park et al. (2016) extends the task itself to include generating explanations for produced answers. Chandrasekaran et al. (2017) take a different approach, asking how well humans can learn patterns in answers and failures of a trained VQA model. While humans indeed identify some patterns, they do not gain any apparent insight from knowing intermediate states of the model (such as its attentional focus). In light of this, we are motivated by the goal of developing more transparent AI.

We address this at the level of counting in VQA. We show that, despite the challenge presented by this particular task, an intuitive approach gains in both performance and interpretability over state of the art.

3 Datasets

Within the field of VQA, the majority of progress has been aimed at the VQA dataset (Agrawal et al., 2015) and, more recently, VQA 2.0 (Goyal et al., 2017), which expands the total number of questions in the dataset and attempts to reduce bias by balancing answers to repeated questions. VQA 2.0 consists of 1.1M questions pertaining to the 205K images from COCO (Lin et al., 2014). The examples are divided according to the official COCO splits.

In addition to VQA 2.0, we incorporate the Visual Genome (VG) dataset (Krishna et al., 2016). Visual Genome consists of 108K images, roughly half of which are part of COCO. Annotations for each image include: (i) Scene objects: bounding box coordinates, object class, and object attributes (ii) Relationships between objects: subject/predicate/object triplets (iii) Region descriptions: bounding box coordinates and a short description of its contents and (iv) Question-answer pairs. We incorporate these data in two ways: First, we use the object, relationship, and region description labels to pre-train the Vision module (Sec. 4.1). Second, we augment the VQA 2.0 training data to include the QA pairs from Visual Genome. We do not train on Visual Genome data when the image is part of the VQA 2.0 validation set since we use a separate split of this set for testing.

3.1 HowMany-QA

In order to evaluate counting specifically, we define a subset of the QA pairs, which we refer to as HowMany-QA. Our inclusion criteria were designed to filter QA pairs where the question asks for a count, as opposed to simply an answer in the form of a number (Fig 2). For the first condition, we require that the question contains one of the following phrases: "how many", "number of", "amount of", or "count of". We also reject a question if it contains the phrase "number of the", since this phrase frequently refers to a printed number rather than a count (i.e. "what is the number of the bus?"). Lastly, we require that the ground-truth answer is a number between 0 to 20 (inclusive). The original VQA 2.0 train set includes roughly 444K QA pairs, of which 57,606 are labeled as having a "number" answer. Focusing on counting questions results in a still very large dataset with 47,542 pairs showing the importance of this subtask. We will make the filtering scripts available so future research can compare on this same dataset.

Figure 2: Examples of question-answer pairs that are excluded from HowMany-QA. This selection exemplifies the common types of "number" questions that do not require counting and therefore distract from our objective: (from left to right) time, general number-based answers, ballparking, and reading numbers from images. Importantly, the standard VQA evaluation metrics do not distinguish these from counting questions; instead, performance is reported for "number" questions as a whole.

Due to our filter and focus on counting questions, we cannot make use of the official test data since its annotations are not available. Hence, we divide the validation data into separate development and test sets. More specifically, we apply the above criteria to the official validation data and select 5,000 of the resulting QA pairs to serve as the test data. The remaining 17,714 QA pairs are used as the development set.

Split	QA Pairs	Images
Train	136,232	79,358
from VQA 2.0	47,542	31,932
from VG	88,690	33,812
Dev.	17,714	13,119
Test	5,000	2,483

Table 1: Size breakdown of HowMany-QA. The contributions of VQA 2.0 and Visual Genome are provided for the train split. (Neither development or test included Visual Genome data.)

As mentioned above, the HowMany-QA training data is augmented with available QA pairs from Visual Genome, which are selected using the same criteria. A breakdown of the size and composition of HowMany-QA is provided in Table 1. All models compared in this work are trained and evaluated on HowMany-QA.

4 Model

In this work, we focus specifically on counting in the setting of visual question answering (where the criteria for counting changes on a question-by-question basis). We experiment with three models. These models use identical strategies to encode the image and the question but differ in terms of how those encodings are used to produce a count (Fig. 3). The Vision module serves to identify objects and represent each in a way that is useful for determining its relevance to a phrase in the question (such as "blue car"). This goal is motivated by the fact that each of the models involves comparing the question to each of the detected objects. We begin by describing the common strategy for representing the contents of the image and go on to describe the strategies specific to each architecture.

4.1 Object Detection

Our approach is inspired by the strategy of Anderson et al. (2017). Their model, which represents current state of the art in VQA, infers objects as the input to the question-answering system. We employ a similar approach and use a Region-based Fully Convolutional Network (R-FCN) to perform object detection (Dai et al., 2016), which uses ResNet-101 (He et al., 2016) as its backbone architecture. We define $h = \texttt{CNN}(I)$ to be the final output feature map of the ResNet after processing the image I. The R-FCN outputs a set of bounding boxes $\{b_1, ..., b_N\}$, $b_i \in \mathbb{R}^4$ corresponding to the locations of each of the N detected objects. We extract the average feature activation of h within the boundaries of b_i and use this as the encoding for the corresponding object $v_i \in \mathbb{R}^{2048}$. From these, we treat the representation of the image as the set of encodings for each detected object $\{v_1, ..., v_N\}$.

As we describe in greater detail below, our featured model learns to make use of inferred relationships between objects when counting. In particular, the relationships shape how the decision to count one object influence subsequent decisions of which object to count. In order to use of this information, our vision module classifies common relationships between pairs of objects, which are computed as a function of the pair's coordinates (normalized to $[-1, 1]$) and region codes.

$$p_{ij}^R = \text{softmax}\left(f^R\left([b_i, b_j, v_i, v_j]\right)\right) \tag{1}$$

where $f^R : \mathbb{R}^m \rightarrow \mathbb{R}^7$ is a two-layer MLP with ReLU activations. $[x, y]$ denotes concatenation of x and y.

We focus on 6 common relationships, which capture basic spatial arrangement (i.e. *on*) and belonging (i.e. *has*).

We perform object detection and collect $\{v_1, ..., v_N\}$, $\{b_1, ..., b_N\}$, and $p^R \in \mathbb{R}^{N \text{x} N}$ for each image in the dataset before training for QA. Each QA model makes use of the same visual representations, which are not fine-tuned during QA training.

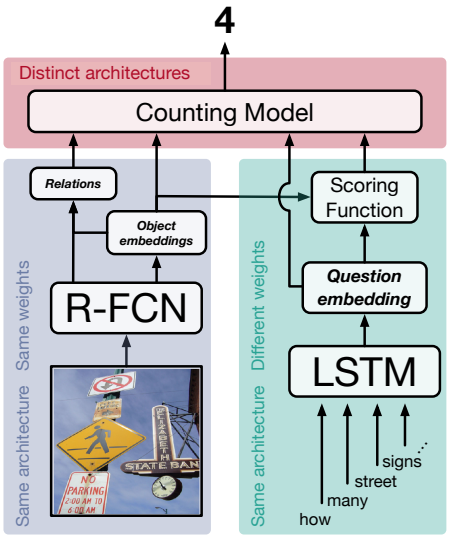

Figure 3: The counting task is built from three basic modules: vision (blue), language (green), and counting (red). Text in the shaded regions describes which aspects of these modules are shared across models.

Training. In order to train the vision module to produce rich object representations, we also train on attribute classification and caption grounding. In their original paper Anderson et al. (2017) did not train on relationship classification or caption grounding; those additions are specific to this work. Training and implementation details are found in the Appendix (Sec. B.1).

4.2 Counting

This work explores three classes of models to perform counting in visual question answering. When answering a question, each architecture begins the QA process by encoding the question and comparing it against each detected object via a scoring function. We define q as the final hidden state of an LSTM (Hochreiter & Schmidhuber, 1997) after processing the question and compute a score vector for each object (Fig. 4):

$$h^t = \text{LSTM}\left(x^t, h^{t-1}\right) \quad q = h^T \tag{2}$$

$$s_i = f^S\left([q, v_i]\right) \tag{3}$$

Here, x_t denotes the word embedding of the question token at position t and $s_i \in \mathbb{R}^n$ denotes the score vector of object i. Following Anderson et al. (2017), we implement the scoring function $f^S : \mathbb{R}^m \rightarrow \mathbb{R}^n$ as a layer of Gated Tanh Units (GTU) (van den Oord et al., 2016).

These same steps occur during caption grounding, giving the option to jointly train the scoring function with the caption grounding and counting objectives (Fig. 4). We randomly initialize the parameters of the scoring function before training on HowMany-QA.

SoftCount. As a baseline approach, we trained a model to count directly from the outputs s of the scoring function. We allow each object to contribute a value between 0 and 1. The total count is the sum of these fractional, object-specific count values. We train this model by minimizing the Huber loss associated with the counting error e:

$$C = \sum_i \sigma\left(W s_i\right) \tag{4}$$

$$L_1 = \begin{cases} 0.5e^2 & \text{if } e \leq 1 \\ e - 0.5 & \text{otherwise} \end{cases} \qquad e = |C - C^{\text{GT}}| \tag{5}$$

For evaluation, we round the estimated count C to the nearest integer.

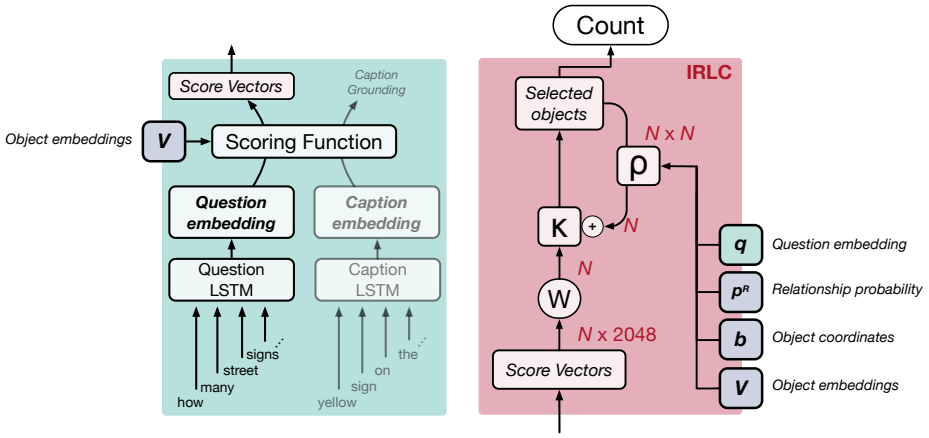

Figure 4: Left: The language model embeds the question and compares it to each object using a scoring function, which is jointly trained with caption grounding for the SoftCount and IRLC models. Right: The counting module of IRLC.

Attention Baseline (UpDown). As a second baseline, we re-implement the QA architecture introduced in Anderson et al. (2017), which the authors refer to as UpDown. We focus on this architecture for three main reasons. First, it represents the current state of the art for VQA 2.0. Second, it has been shown to work well with the type of visual representation we employ. And, third, it exemplifies the common two-stage approach of (1) deploying question-based attention over image regions (here, detected objects) to get a fixed-length visual representation

$$\alpha = \text{softmax}\left(Ws\right); \quad \hat{v} = \sum_i \alpha_i v_i \tag{6}$$

and then (2) classifying the answer based on this average and the question encoding

$$v' = f^V\left(\hat{v}\right); \quad q' = f^Q\left(q\right) \tag{7}$$

$$p = \text{softmax}\left(f^C\left(v' \otimes q'\right)\right) \tag{8}$$

where $s \in \mathbb{R}^{N\text{x}n}$ denotes the matrix of score vectors for each of the N detected objects and $\alpha \in \mathbb{R}^N$ denotes the attention weights. Here, each function f is implemented as a GTU layer and $\otimes$ denotes element-wise multiplication. For training, we use a cross entropy loss, with the target given by the ground-truth count. At test time, we use the most probable count given by p.

Interpretable RL Counter (IRLC). For our proposed model, we aim to learn *how* to count by learning *what* to count. We assume that each counting question implicitly refers to a subset of the objects within a scene that meet some variable criteria. In this sense, the goal of our model is to enumerate that subset of objects.

To implement this as a sequential decision process, we need to represent the probability of selecting a given action and how each action affects subsequent choices. To that end, we project the object scores $s \in \mathbb{R}^{N\text{x}n}$ to a vector of logits $\kappa \in \mathbb{R}^N$, representing how likely each object is to be counted, where N is the number of detected objects:

$$\kappa = Ws + b \tag{9}$$

And we compute a matrix of interaction terms $\rho \in \mathbb{R}^{N\text{x}N}$ that are used to update the logits κ. The value ρ_{ij} represents how selecting object i will change κ_j. We calculate this interaction from a compressed representation of the question (Wq), the dot product of the normalized object vectors ($\hat{v}_i^{\text{T}}\hat{v}_j$), the object coordinates (b_i and b_j), their overlap statistics (IoU_{ij}, O_{ij}, and O_{ji}), and their predicted relationships (p^R_{ij} and p^R_{ji}, from Eq. 1):

$$\rho_{ij} = f^\rho\left(\left[Wq, \hat{v}_i^{\text{T}}\hat{v}_j, b_i, b_j, \text{IoU}_{ij}, \text{O}_{ij}, \text{O}_{ji}, p^R_{ij}, p^R_{ji}\right]\right) \tag{10}$$

where $f^\rho : x \in \mathbb{R}^m \Rightarrow \mathbb{R}$ is a 2-layer MLP with ReLU activations.

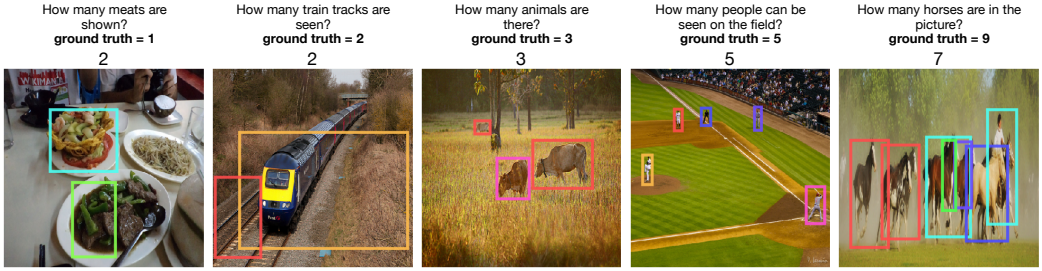

Figure 5: Grounded counts produced by IRLC. Counts are formed from selections of detected objects. Each image displays the objects that IRLC chose to count.

For each step t of the counting sequence we greedily select the action with the highest value (interpreted as either selecting the next object to count or terminating), and update κ accordingly:

$$a^t = \text{argmax}_i \left[\kappa^t, \zeta\right] \tag{11}$$

$$\kappa^{t+1} = \kappa^t + \rho_{a^t} \tag{12}$$

where ζ is a learnable scalar representing the logit value of the terminal action, and κ^0 is the result of Equation 9. Each object is only allowed to be counted once. We define the count C as the timestep when the terminal action was selected $t : a^t = N + 1$.

This approach bears some similarity to Non-Maximal Suppression (NMS), a staple technique in object detection to suppress redundant proposals. However, our approach is far less rigid and allows the question to determine which types of relationships between objects control their interaction.

Training IRLC. Because the process of generating a count requires making discrete decisions, training requires that we use techniques from Reinforcement Learning. Given our formulation, a natural choice is to apply REINFORCE (Williams, 1992). To do so, we calculate a distribution over action probabilities p^t from κ^t and generate a count by iteratively sampling actions from the distribution:

$$p^t = \text{softmax}\left(\left[\kappa^t, \zeta\right]\right) \quad a^t \sim p^t \tag{13}$$

$$\kappa^{t+1} = \kappa^t + \rho_{a^t} \tag{14}$$

We calculate the reward using Self-Critical Sequence training (Rennie et al., 2017; Anderson et al., 2017; Paulus et al., 2017), a variation of policy gradient. We define $E = |C - C^{\text{GT}}|$ to be the count error and use $R = E - E^{\text{greedy}}$, where E^{greedy} is the baseline count error obtained by greedy action selection (which is also how the count is measured at test time). From this, we define our (unnormalized) counting loss as

$$\tilde{L}_C = -R \sum_t \log p^t\left(a^t\right) \tag{15}$$

Additionally, we include two auxiliary objectives to aid learning. For each sampled sequence, we measure the total negative policy entropy H across the observed time steps. We also measure the average interaction strength at each time step and collect the total

$$\tilde{P}_{\text{H}} = -\sum_t H\left(p^t\right) \qquad \tilde{P}_{\text{I}} = \sum_{i \in \{a^0 \ldots a^t\}} \frac{1}{N} \sum_j L_1\left(\rho_{ij}\right) \tag{16}$$

where L_1 is the Huber loss from Eq 5. Including the entropy objective is a common strategy when using policy gradient (Williams & Peng, 1991; Minh et al., 2016; Luo et al., 2017) and is used to improve exploration. The interaction penalty is motivated by the *a priori* expectation that interactions should be sparse. From our observations, both terms significantly influence the strategy the model ultimately learns to use. During training, we minimize a weighted sum of the three losses, normalized by the number of decision steps. As before, we provide training and implementation details in the Appendix (Sec. B.2).

	Accuracy						RMSE					
	Common		⟶		Unseen		Common		⟶		Unseen	
SoftCount	52.8	51.5	43.4	45.2	40.1	38.6	2.27	2.41	2.87	2.92	3.28	3.30
UpDown	55.8	54.8	47.3	47.2	**43.9**	39.0	2.50	2.43	3.03	3.06	3.31	3.32
IRLC	**57.9**	**56.1**	**49.7**	**48.1**	42.7	**40.6**	**2.20**	**2.24**	**2.69**	**2.85**	**3.21**	**3.26**

Table 3: Model performance grouped according to the frequency with which the counting subject appeared in the training data. Metrics are reported for each of the 6 frequency bins. For each metric, the data are organized such that the most common subjects contribute to the leftmost bin.

5 RESULTS

5.1 COUNTING PERFORMANCE

We use two metrics for evaluation. For consistency with past work, we report the standard VQA test metric of accuracy. Since accuracy does not measure the degree of error we also report root-mean-squared-error (RMSE), which captures the typical deviation between the estimated and ground-truth count and emphasizes extreme errors. Details are provided in the Appendix (Sec. C).

Model	Accuracy	RMSE
SoftCount	46.6	2.61
UpDown	47.4	2.86
IRLC	**49.7**	**2.51**

Table 2: HowMany-QA test set performance

IRLC achieves the highest overall accuracy and lowest overall RMSE on the test set (Table 2). Interestingly, SoftCount clearly lags in accuracy but is competitive in RMSE. This suggests that SoftCount is usually able to get closer to the correct count than UpDown but more frequently fails to identify the exact count. Furthermore, it argues that accuracy and RMSE and not redundant. Therefore, we emphasize that IRLC achieves the best performance for both metrics.

To gain more insight into the performance of these models, we calculate these metrics within the development set after separating the data according to how common the subject of the count is during training[1]. We break up the questions into 5 roughly equal-sized bins representing increasingly uncommon subjects. We include a 6th bin for subjects never seen during training. The accuracy and RMSE across the development set are reported for each of these bins in Table 3.

Organizing the data this way reveals two main trends. First, all models perform better when asked to count subjects that were common during training. Second, the performance improvements offered by IRLC over UpDown persist over nearly all of groupings of the development data.

5.2 QUALITATIVE ANALYSIS

The design of IRLC is inspired by the ideal of interpretable VQA (Chandrasekaran et al., 2017). One hallmark of interpretability is the ability to predict failure modes. We argue that this is made more approachable by requiring IRLC to identify the objects in the scene that it chooses to count.

Consider the effect of subject frequency during training (Table 3). We can attempt to understand the poorer performance with rare subjects by comparing the counting patterns within different groups and overall, as in Figure 6. After inspection, we can take away that performance falls off as the target count gets higher and that this trend worsens with the rarity of the counting subject. In all likelihood, this reflects the fact that the training data are dominated by small ground-truth counts and that, when the subject is unfamiliar, the models resort to learned biases. Unfortunately, this is not particularly surprising and these holistic trends only weakly inform what each model has/has not learned. We do gain such insight, however, when examining the grounded outputs of IRLC.

[1]We infer the subject using a simple dependency-parsing heuristic. Specifically, we extract the root word of the first noun-chunk.

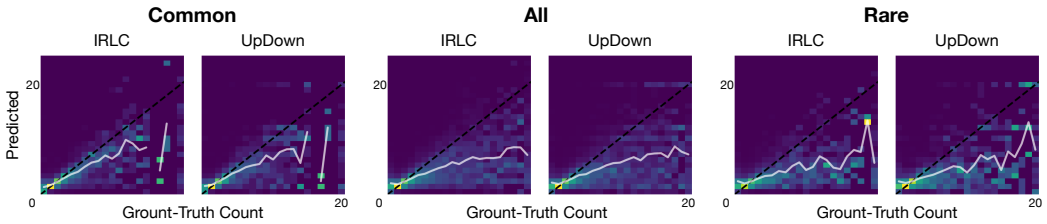

Figure 6: Histograms of predicted count, given the ground-truth count. Histograms are plotted for the most common quintile of training subjects (left), the least common (right), and the entire development set (middle). Columns are normalized to account for imbalance in ground-truth counts. White lines are the average predicted count at each ground-truth count (perfect performance is indicated by the line of unity, shown in black). IRLC is able to produce counts greater than 20. For both models, under-counting is common. This tendency is more severe with rare counting subjects, as the models resort to the learned prior of small ground truth counts.

Figure 7: Examples of failure cases with common and rare subjects. Each example shows the output of IRLC, where boxes correspond to counted objects, and the output of UpDown, where boxes are shaded according to their attention weights (Eq. 6).

Figure 7 illustrates two failure cases that exemplify observed trends in IRLC. In particular, IRLC has little trouble counting people (they are the most common subject) but encounters difficulty with referring phrases (in this case, "sitting on the bench"). When asked to count burners (a rare subject), the IRLC focuses on the wrong type of object (the control knobs). These failures are obvious by virtue of the grounded counts, which point out exactly which objects IRLC counted. In comparison, the attention focus of UpDown (representing the closest analogy to a grounded output) does not identify any pattern. From the attention weights, it is unclear which scene elements form the basis of the returned count.

Indeed, the two models may share similar deficits. We observe that, in many cases, they produce similar counts. However, we stress that, without IRLC and the chance to observe such similarities, such deficits of the UpDown model would be difficult to identify.

The Appendix includes further visualizations and comparisons of model output, including examples of how IRLC uses the iterative decision process to produce discrete, grounded counts (Sec. A).

6 Conclusion

We present an interpretable approach to counting in visual question answering, based on learning to enumerate objects in a scene. By using RL, we are able to train our model to make binary decisions about whether a detected object contributes to the final count. We experiment with two additional baselines and control for variations due to visual representations and for the mechanism of visual-linguistic comparison. Our approach surpasses both baselines for each of our evaluation metrics. In

addition, our model identifies the objects that contribute to each count. These groundings provide traction for identifying the aspects of the task that the model has failed to learn and thereby improve not only performance but also interpretability.

REFERENCES

Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Devi Parikh, and Dhruv Batra. VQA: Visual Question Answering. *International Journal of Computer Vision*, 2015.

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 VQA. *arXiv*, 2017.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *CVPR*, 2016a.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to Compose Neural Networks for Question Answering. In *NAACL*, 2016b.

Arjun Chandrasekaran, Deshraj Yadav, Prithvijit Chattopadhyay, Viraj Prabhu, and Devi Parikh. It Takes Two to Tango: Towards Theory of AI's Mind. *arXiv*, 2017.

Prithvijit Chattopadhyay, Ramakrishna Vedantam, Ramprasaath R. Selvaraju, Dhruv Batra, and Devi Parikh. Counting Everyday Objects in Everyday Scenes. In *CVPR*, 2017.

Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. In *NIPS*, 2016.

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. In *EMNLP*, 2016.

Ross Girshick. Fast R-CNN. In *ICCV*, 2015.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2015.

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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to Reason: End-to-End Module Networks for Visual Question Answering. In *ICCV*, 2017.

Ilija Ilievski, Shuicheng Yan, and Jiashi Feng. A Focused Dynamic Attention Model for Visual Question Answering. *arXiv*, 2016.

Jin-Hwa Kim, Kyoung-Woon On, Woosang Lim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Hadamard Product for Low-rank Bilinear Pooling. In *ICLR*, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv*, 2014.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. *International Journal of Computer Vision*, 2016.

Victor Lempitsky and Andrew Zisserman. Learning To Count Objects in Images. *NIPS*, 2010.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. In *ECCV*, 2014.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. *ICCV*, 2017.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning. In *CVPR*, 2016a.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical Question-Image Co-Attention for Visual Question Answering. In *NIPS*, 2016b.

Yuping Luo, Chung-cheng Chiu, Navdeep Jaitly, and Ilya Sutskever. Learning Online Alignments with Continuous Rewards Policy Gradient. In *ICASSP*, 2017.

Mateusz Malinowski and Mario Fritz. A Multi-World Approach to Question Answering about Real-World Scenes based on Uncertain Input. In *NIPS*, 2014.

Volodymyr Minh, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. In *ICML*, 2016.

Daniel Oñoro-Rubio and Roberto J. López-Sastre. Towards perspective-free object counting with deep learning. In *ECCV*, 2016.

Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. Attentive Explanations: Justifying Decisions and Pointing to the Evidence. *arXiv*, 2016.

Romain Paulus, Caiming Xiong, and Richard Socher. A Deep Reinforced Model for Abstractive Summarization. *arXiv*, 2017.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global Vectors for Word Representation. *EMNLP*, 2014.

Mengye Ren and Richard S. Zemel. End-to-End Instance Segmentation with Recurrent Attention. In *CVPR*, 2017.

Mengye Ren, Ryan Kiros, and Richard Zemel. Exploring Models and Data for Image Question Answering. In *NIPS*, 2015a.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NIPS*, 2015b.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical Sequence Training for Image Captioning. In *CVPR*, 2017.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 2015.

Santi Segui, Oriol Pujol, and Jordi Vitria. Learning to count with deep object features. In *CVPRW*, 2015.

Kevin J. Shih, Saurabh Singh, and Derek Hoiem. Where To Look: Focus Regions for Visual Question Answering. In *CVPR*, 2015.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. Grounded Compositional Semantics for Finding and Describing Images with Sentences. In *TACL*, 2014.

Damien Teney, Lingqiao Liu, and Anton van den Hengel. Graph-Structured Representations for Visual Question Answering. *arXiv*, 2016.

Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional Image Generation with PixelCNN Decoders. In *NIPS*, 2016.

R J Williams. Simple statistical gradient-following methods for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Ronald J. Williams and Jing Peng. Function Optimization using Connectionist Reinforcement Learning Algorithms. *Connection Science*, 3(3):241–268, 1991.

Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic Memory Networks for Visual and Textual Question Answering. In *ICML*, 2016.

Huijuan Xu and Kate Saenko. Ask, Attend and Answer: Exploring Question-Guided Spatial Attention for Visual Question Answering. In *ECCV*, 2015.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked Attention Networks for Image Question Answering. In *CVPR*, 2015.

Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *CVPR*, 2015.

Jianming Zhang, Shugao Ma, Mehrnoosh Sameki, Stan Sclaroff, Margrit Betke, Zhe Lin, Xiaohui Shen, Brian Price, and Radomír Měch. Salient Object Subitizing. *International Journal of Computer Vision*, 2017.

Bolei Zhou, Yuandong Tian, Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Simple Baseline for Visual Question Answering. *arXiv*, 2015.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. Visual7W: Grounded Question Answering in Images. In *CVPR*, 2015.

A EXAMPLES

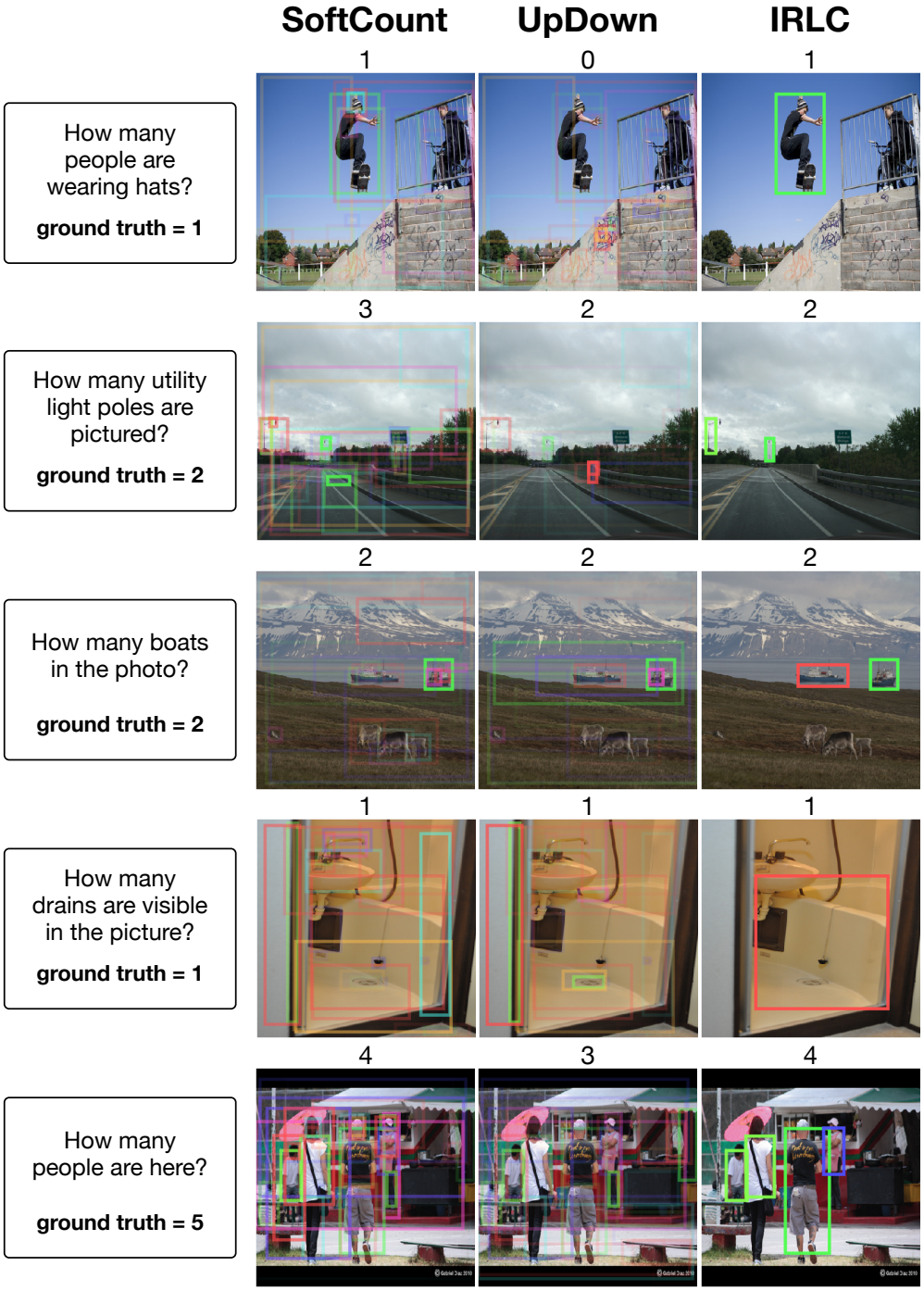

Figure 8: Example outputs produced by each model. For SoftCount, objects are shaded according to the fractional count of each (0=transparent; 1=opaque). For UpDown, we similarly shade the objects but use the attention focus to determine opacity. For IRLC, we plot only the boxes from objects that were selected as part of the count.

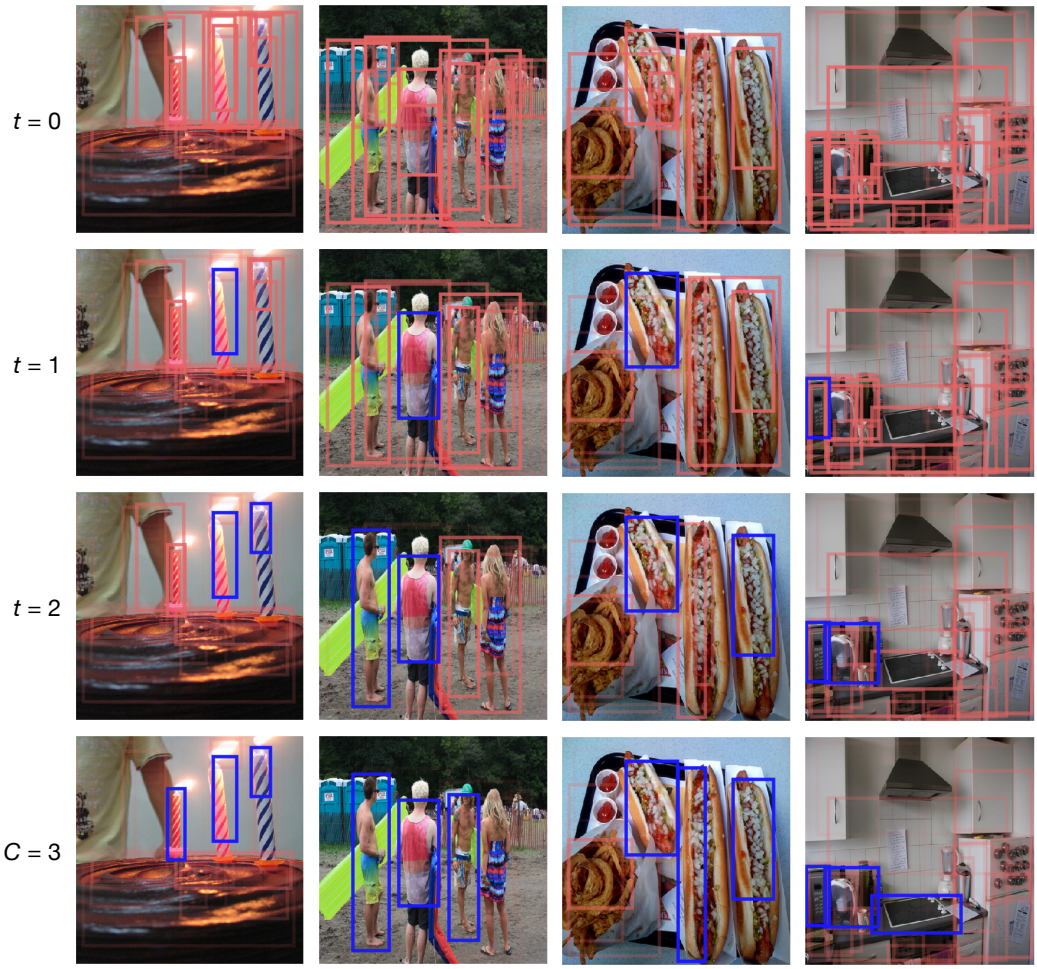

Figure 9: Sequential counting of IRLC. At each timestep, we illustrate the unchosen boxes in pink, and shade each box according to κ^t (corresponding to the probability that the box would be selected at that time step; see main text). We also show the already-selected boxes in blue. For each of the questions, the counting sequence terminates at $t = 3$, meaning that the returned count C is 3. For each of these questions, that is the correct answer. The initial scores are lack precision but are sufficient for the sequential decision process to identify the set of distinct objects that correspond to the subject of the question.

B Training and Implementation Details

B.1 Vision Module

Our vision module is trained towards 5 basic tasks: bounding box regression, classification of detected objects, classification of object attributes, classification of pairwise relationships between objects, and grounding captions in the objects they describe. We train using the labels provided in the Visual Genome dataset (Krishna et al., 2016). We limit training to objects belonging to one of the 1200 most common classes. We make use of the 500 most common attributes and the 6 most common relationship types. Finally, it was necessary to resolve the different bounding boxes given to objects and those given to region captions. We apply a greedy heuristic based on Intersection over

Union to identify valid assignments, such that only a subset of objects were given associated captions. An interesting consequence of this is that many captions make reference to objects' context within the whole image, often amounting to a linguistic description of a labeled relationships.

We follow existing methods for training CNNs for object detection, which serve to train bounding box regression and classification of object type (Dai et al., 2016). The final outputs of the R-FCN serve as the inputs to the additional auxiliary tasks, so we do not require any reformulation of the original approach beyond the addition of a few extra steps.

In the main text, we refer to object encodings v and bounding boxes b as estimated during inference. For training, we assign ground-truth boxes to one of many anchor locations of the R-FCN and use the boxes and encodings ultimately produced from that location. Therefore, in the following descriptions, v and b refer specifically to features associated with some ground-truth object. For attribute classification and caption grounding, we form a batch from all such positive regions.

Attribute classification. We learn to classify labeled attributes from the concatenation of the region code and a learned embedding of the ground-truth class $y_i \in \mathbb{R}^{128}$.

$$p_i^A = \text{softmax}\left(f^A\left([y_i, v_i]\right)\right), \tag{17}$$

where $f^A : \mathbb{R}^m \to \mathbb{R}^{501}$ is a multi-layer perceptron (MLP) with ReLU activations.

Up to this point, our strategy closely mirrors the state of the art approach from Anderson et al. (2017). The main differences begin with our choice to also train relationship classification (which is described in the main text) and caption grounding.

Caption grounding. The goal of caption grounding is, given a set of objects and a caption, to identify the object that the caption describes. We use an LSTM to encode the caption and compare it to each of the objects:

$$h_i^t = \text{LSTM}\left(x_i^t, h_i^{t-1}\right) \tag{18}$$

$$\alpha_{ij} = W f^S\left(\left[h_i^T, v_j\right]\right) + b \tag{19}$$

$$p_{ij} = \text{softmax}\left(\alpha_i\right)_j \tag{20}$$

where $h \in \mathbb{R}^{1024}$, $x_i^t \in \mathbb{R}^{300}$ is the embedding for the token at timestep t of the caption indexed by i, T is the caption length, and $f^S : \mathbb{R}^m \to \mathbb{R}^n$ is the scoring function (Sec. 4.2). $\alpha_{ij} \in \mathbb{R}$ is the score of the pair formed by caption i and object j. We only score the subset of objects that have a caption and train p_{ij} to be largest when $i = j$.

Rather than train on all possible object pairs, we create a batch out of all the pairs with a labeled relationship and pad with randomly sampled no-relationship pairs to reach a batch size of 128 pairs per image. We initialize the ResNet using weights pre-trained on ImageNet (Russakovsky et al., 2015). We train all tasks end-to-end following one epoch of only training towards the traditional R-FCN objectives. We use SGD+Momentum with a learning rate of 0.0001, which we decay each iteration by a factor of 0.99997. Word vectors were initialized from Glove embeddings (Pennington et al., 2014) and fine-tuned during caption matching. We regularize using dropout and early stopping.

Inference. At test time, we perform object detection using standard procedure. We identify 300 candidate regions from the RPN (Ren et al., 2015a) after performing Non-Maximal Suppression with an IoU threshold of 0.5. We then complete the R-FCN pipeline and perform class-specific NMS with a threshold of 0.3, taking the top 100 results. Lastly, we filter any detected objects with an estimated background probability ≥ 0.2.

B.2 Counting models

Each of the considered counting models makes use of the same basic architecture for encoding the question and comparing it with each of the detected objects. For each model, we initialized the word embeddings to those learned from caption grounding during pre-training and encoded the question with an LSTM of hidden size 1024. The only differences in the model-specific implementations concern the hidden size of the scoring function f^S and joint-training with caption grounding. We determined these specifics from the optimal settings observed during initial experiments. We use a hidden size of 512 for SoftCount and UpDown and a hidden size of 2048 for IRLC. We observed that

the former two models were more prone to overfitting, whereas IRLC benefited from the increased capacity.

We include joint training of caption grounding for SoftCount and IRLC. With IRLC, we also tie the weight vector W that projects the score vectors during counting and caption grounding. During training, we randomly select four of the images in the batch of examples to use for caption grounding (rather than the full 32 images that make up a batch). Caption grounding with detected boxes also requires that we assign predicted detections with ground-truth boxes. To do so, we select assignments greedily using an IoU threshold of 0.5. We weight the loss associated with caption grounding by 0.1 relative to the counting loss. We did not perform any joint training with UpDown, since the two objectives appeared to interfere with one another during initial experiments.

When training on counting, we optimize using Adam (Kingma & Ba, 2014). For SoftCount and UpDown, we use a learning rate of $3\text{x}10^{-4}$ and decay the learning rate by 0.8 when the training accuracy plateaus. For IRLC, we use a learning rate of $5\text{x}10^{-4}$ and decay the learning rate by 0.99999 every iteration. For all models, we regularize using dropout and apply early stopping based on the Development set accuracy (see below).

When training IRLC, we apply the sampling procedure 5 times per question and average the losses. We weight the entropy penalty P_H and interaction penalty P_I (Eq. 16) by 0.002 and 0.01, respectively, relative to the counting loss.

C EVALUATION METRICS

Accuracy. The VQA dataset includes annotations from ten human reviewers per question. The accuracy of a given answer a depends on how many of the provided answers it agrees with. It is scored as correct if at least 3 humans answers agree: $\texttt{Acc}\left(a\right) = \texttt{min}\left[\frac{\text{\#humans that said } a}{3}, 1\right]$. Each answer's accuracy is averaged over each 10-choose-9 set of human answers. As described in the main text, we only consider examples where the consensus answer was in the range of 0-20. We use all ten labels to calculate accuracy, regardless of whether individual labels deviate from this range. We accuracy values we report are taken from the average accuracy over some set of examples.

RMSE. This metric simply quantifies the typical deviation between the model count and the ground-truth. Across a set of N, we calculate this metric as $\texttt{RMSE} = \sqrt{\frac{1}{N}\sum_i (\hat{C}_i - C_i)^2}$, where $\hat{C}_i$ and C_i are the predicted and ground-truth counts, respectively, for question i. RMSE is a measurement of error, so lower is better.