Counting Everyday Objects in Everyday Scenes

Prithvijit Chattopadhyay*, Ramakrishna Vedantam*, Ramprasaath RS, Dhruv Batra, and Devi Parikh

Virginia Tech
{prithv1,vrama91,ram21,dbatra,parikh}@vt.edu

Abstract. We introduce the problem of counting *everyday* objects in *everyday* scenes. While previous works have studied specific counting problems such as pedestrian counting in surveillance videos, or biological cell counting, we are interested in counting common objects in natural scenes. We study this problem in a setup similar to traditional scene understanding problems. Given an image, we consider the task of predicting the counts (or the numerosity) of categories of interest. We study some simple approaches and applications for this counting problem. Our `detect` approach adapts an object detector to perform counting, while our `glance` approach regresses to ground truth counts. Our associative subitizing (`aso-sub`) approach divides an image into regions and regresses to fractional object counts in each region. We create an ensemble (`ens`) of these counting methods which improves performance. We demonstrate counting performance on the PASCAL and MS COCO datasets. We show proof-of-concept applications of our automatic counting methods to 1) improve object detection performance, and 2) visual question answering (on VQA and COCO-QA). Our code and datasets will be publicly available.

1 Introduction

We study the problem of counting everyday objects in everyday scenes. The ability to count is one of the essential tools we use to navigate through our everyday lives. Thus, learning to count objects has applications in building smarter digital assistants, which interact seamlessly with humans.

Moreover, counting common objects in natural scenes is an interesting scene understanding problem. There is a rich history of work in detection [1,2,3,4] and semantic segmentation [5,6] which aim to reason about finer details in the image than the image classification [7,8]. Counting is an orthogonal fine-grained scene understanding task. Unlike image classification, where the task is to indicate the presence or absence of an object, counting requires one to reason *how many* instances of an object are present in the scene. Interestingly, counting also has connections to recent efforts on VQA [9,10], where common question categories ("how many?") require reasoning about counts.

Counting by Detecting: How does counting relate to popular scene understanding tasks like detection? It is easy to realize that perfect detection of objects

* Equal contribution

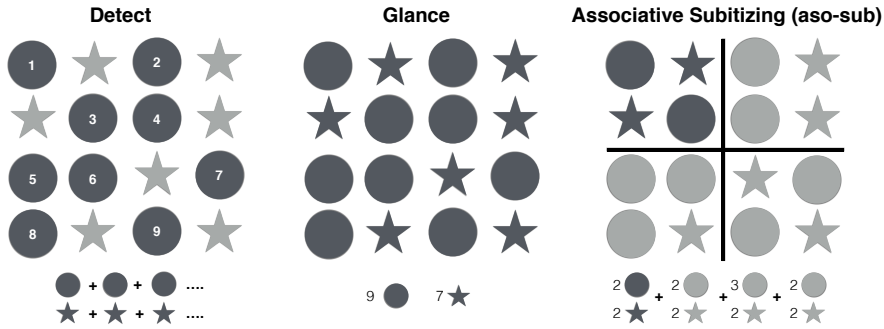


Fig. 1: A toy example explaining the motivation for counting approaches explored in this paper. The task is to count the number of stars and circles. In `detect`, the idea is to detect instances of a category, and then report the total number of instances detected as the count. In `glance`, we make a judgment of count based on a glimpse of the full image. In `aso-sub`, we divide the image into regions and judge count based on patterns in local regions. Counts from different regions are added through arithmetic.

would also lead to a perfect count. Thus, detection is clearly sufficient for counting. However, the counting problem does not require localization. For instance, for counting mugs, we need not determine (with pixel-accurate segmentations or even bounding boxes) where exactly they are, as long as we are able to determine how many handles exist. Thus, it might not be necessary to detect objects accurately in order to count. However, detection still forms a natural approach for counting. In this paper, we use the Fast R-CNN detector [11] for "counting by detecting" (`detect`). Note that this approach, however has not explicitly learnt to count.

Counting by Glancing: Representations extracted from Deep Convolutional Neural Networks [12,7] trained on image classification have been successfully applied to a number of scene understanding tasks such as finegrained recognition [13], scene classification [13], object detection [13] *etc.*. Thus, another natural approach for counting is to use the features extracted from a deep network and regress to counts. Unlike detection, such a method explicitly *learns to count*. In a sense, this method estimates counts from a glance of the entire scene (as opposed to `detect` which first localizes objects). We call this method "counting by glancing" or `glance`.

Counting by Associative Subitizing: There is rich literature in pedagogy and psychology on how children acquire skills of counting [14,15]. Children are known to have an ability to map a perceptual signal to an estimate of number (usually accurate for smaller numbers) [16]. This phenomenon, called subitizing has been thought to be a developmental prerequisite to counting [17]. Thus counting, subitizing and arithmetic are closely linked.

We adapt these ideas to our problem (Fig. 1). One can think of the `glance` method as an attempt to subitize using the whole image. However, as shown in Fig. 1, subitizing by `glance` can be difficult when the number of instances of an object in a region is large. To resolve this, we divide the image into non-

overlapping cells (Fig. 1 right). We then subitize in each cell and use addition to get the total count. We call this method associative subitizing or `aso-sub`.

One can think of `aso-sub` as exploring a continuum between `glance` and `detect`. At one end of the spectrum is the special case of detection, where one can only make binary decisions on object count, which can then be summed up over the entire image to get image level counts. On the other end of the spectrum is glancing, which can be thought of as a trivial split of the image (into the image itself), that is, subitizing at the entire image level.

Counting by Ensembling: It is well known that when humans are given counting problems with large ground truth counts (*e.g.* counting number of pebbles in a jar), individual guesses have high variance, but an average across multiple responses tends to be surprisingly close to the ground truth. Inspired by this, we create an ensemble of counting methods (`ens`). Interestingly, we find that `ens` improves performance over individual counting methods.

Contributions: We introduce the problem of counting everyday objects in everyday scenes, as a novel scene understanding problem. We present four methods for this task: counting by detection (`detect`), counting by glancing (`glance`), counting by associative subitizing (`aso-sub`) and counting by ensembling (`ens`). We then study applications of this counting task. Firstly, we investigate how counting can help detection. We realize that detectors when used in practice have to be run at some operating point on a precision-recall curve. This operating point is typically tuned on a per-category basis. However, if we had an estimate of how many objects were present in the image, we could then set the threshold on a per-image basis to detect that many objects. We also explore counting questions ("how many?") in the Visual Question Answering (VQA) [9] and COCO-QA [10] datasets and provide some preliminary results.

The rest of the paper is organized as follows. We discuss previous work related to counting in Section. 2. We then discuss our approaches to counting in Section. 3. We then detail our experimental setup in Section. 4. We compare various counting methods and present applications of counting in Section. 5.

2 Related Work

We first discuss the state-of-the-art in detection. We then discuss previous work on specialized counting problems such as counting pedestrians in surveillance videos, counting cells in microscopic images, *etc.*.

Object Detection: Recent years have seen a lot of progress in object detection driven by deep learning. R-CNN [3] treats object detection as a classification problem over object proposal windows [18,19,20]. Fast R-CNN [11] improves over R-CNN by sharing computation across overlapping windows by adding a Region of Interest (ROI) pooling layer into the CNN architecture. We use this method with hyperparameter tuning (Section. 3) for counting (`detect`). Other methods for detection include [21,22,23]. In contrast to these approaches, we are interested in counting instances of an object, and not in precise localization. In fact, we find that if our goal is to estimate counts, we can modify existing models to *trade off* the ability to localize with the ability to count, which is a novel finding about deep models.

Counting: Counting people in crowded scenes is a problem with rich literature in computer vision [24,25,26,27] with applications in surveillance. [26] explores a Bayesian Poisson regression method on low-level features for counting crowds. [28] segments a surveillance video into components of homogeneous motion and regresses to counts in each region using Gaussian process regression. Since surveillance scenes tend to be crowded, counting by detection is infeasible due to large amounts of occlusion. Lempitsky and Zisserman [27] avoid detecting people for counting by estimating object density using low-level features. They show applications on surveillance and cell counting in biological images. Zhang *et.al.* [24] train a CNN to count people by predicting per-patch density maps and counts jointly for regions in a scene using `conv3` features. The key difference w.r.t our work is that all previous methods approach this problem in constrained settings (*e.g.* count people/cells with images from a fixed camera) where low-level visual cues are indicative of object count. In contrast, our work counts common objects ('dogs', 'mugs', 'chairs', *etc.*), in natural scenes, where we need to deal with the large intra- and inter-class variance found in object categories, while capturing semantics. Indeed, we find that a baseline approach using `conv3` features does not perform favourably on our counting task. Anchovi labs provided users interactive services to count specific objects such as swimming pools in satellite images, cells in biological images, *etc.*. Recent work [29] explores CNNs to perform salient object subitizing. That is, count the number of salient objects in the image irrespective of what class they belong to. In contrast, we are interested in counting the number of instances of objects per category.

3 Approach

We briefly explain our problem setup. We then describe and motivate our `detect` and `aso-sub` methods. We discuss the architecture choices for `glance` and `aso-sub` (Fig. 2). **Setup:** We assume that we have a dataset of images with counts annotated for each category. The task at test time is to count the number of objects of each category given an image. For training, we need counts of all object categories for each image (for `glance`) and ground truth object bounding boxes (for `detect` and `aso-sub`). The count predictions from the models are compared to the ground truth image-level counts using the mean - root mean squared error (mMSE) (Section. 4.2).

3.1 Counting by Detection (`detect`)

We use the Fast R-CNN [11] detector to count. Detectors typically perform two post processing steps on a set of preliminary boxes (and scores): non maximal suppression (NMS) and score thresholding. NMS discards highly overlapping, and likely redundant detections, whereas the score threshold weeds out all detections with low scores. The NMS step also has a parameter to control the overlap to decide which boxes should be considered redundant.

We steer the detector to count better by varying these two hyperparameters to find the setting where counting error is the least. We pick these parameters

using grid search on a held-out VAL set. We first select a fixed NMS threshold for all the classes and vary the score threshold between 0 and 1. We then vary the NMS threshold for all the classes between 0 to 1 and pass all resulting boxes through a fixed score threshold.

Associative Subitizing (`aso-sub`) The idea of the `aso-sub` is to adapt subitizing [16] for our counting problem. However, unlike the toy example in Fig. 1, objects in real images occur at different scales and positions. Thus, a single object might be split across multiple cells. Hence, we use partial counts denoting the extent of an object category (across instances) in a grid as the ground truth. Conceptually, adapting subitizing to use partial counts would still capture meaning and work well in practice since the CNN representation is distributed [30]. The key intuition for `aso-sub` then is as follows. At some canonical scale, we would be able to count the smaller objects more easily by subitizing them, as well as predict the partial counts for larger objects. In practice, we divide the image into cells (Fig. 2) and regress to counts in each cell using CNN features. The predicted (partial) counts across regions are then summed up independently for each category to produce the final counts. We threshold the count predictions at 0 before summing them across cells.

Given the detection ground truth bounding boxes $c = \{c_1, \cdots, c_n\}$ for a category c, we compute the *target* partial counts (S^c_{gt}) for each grid-cell S:

$$S^c_{gt} = \sum_{i=1}^{n} \frac{S \cap c_i}{c_i} \tag{1}$$

Image Grids: We experiment by dividing the image into equally sized 3×3, 5×5, and 7×7 grid-cells. Here we can form connections between `aso-sub`, `detect`, and `glance`. One can think of `glance` as a trivial 1×1 split on the image, whereas `detect` is a potentially overlapping split of the image into smaller object-like regions. By changing the discretization of the grid, we can loosely interpolate between `detect` and `glance`. We also explored more sophisticated methods involving object density priors, *etc.*, to perform image splitting, but found that this simple split into regular grids works surprisingly well in practice.

We next discuss the architectures we considered for `glance` and `aso-sub`.

3.2 Architectures for counting by glancing (`glance`) and associative-subitizing (`aso-sub`)

An overview of the architectures and problem setup for `glance` and `aso-sub` are given in Fig. 2. For both `glance` and `aso-sub`, our architecture is a multi-layered perceptron (MLP) with batch normalization [31] and Rectified Linear Units (ReLU) activations between hidden layers. We experiment with choices of a single hidden layer, and two hidden layers for the MLP, as well as the sizes of the hidden units. Exact details and performance comparisons can be found in the appendix. The output of the MLP is a vector indicating the predicted counts for each category. We use mean squared error (MSE) between the predictions and

the ground truth counts as the loss to train the MLP. The input to the MLP are `fc7` features from a VGG-16 [12] model. We experiment using both off-the-shelf classification weights from ImageNet [8] and the detection fine-tuned weights from Section. 3.1. We also tried finetuning the CNN for the counting task but found that learning saturates very quickly when finetuning the deeper networks for the regression objective (both MSE and Huber loss). For optimization, we use RMSProp [32] with a minibatch size of 64. The learning rate is set to 0.01 and weight decay is set to 0.01. Validation loss is computed on the mRMSE metric (Sec. 4.2) and the best performing models are used.

4 Experimental Setup

We first describe the datasets and implementation choices for each method while training. We then discuss evaluation metrics for counting. Finally, we explain details of the methods and baselines used for counting.

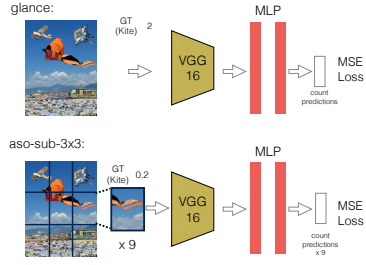


Fig. 2: Architectures used for `glance` and `aso-sub`. The VGG16 [12] is used as a feature extractor and fed into a multi-layered perceptron (MLP).

4.1 Training Data

We learn to count on two datasets depicting everyday objects in everyday scenes: the PASCAL VOC 2007 [33] and MS COCO [34]. Both the datasets contain instance-level detection annotations. We use the ground truth detection annotations to obtain ground truth counts for each image, and each grid for `aso-sub`. The PASCAL VOC dataset contains a TRAIN set of 2501 images, VAL set of 2510 images and a TEST set of 4952 images, and has 20 object categories. The COCO dataset contains a TRAIN set of 82783 images and a VAL set of 40,504 images, with 80 object categories. We use the first half of VAL as a **Count-Val** set and the second half of VAL as the **Count-Test** set. The most frequent count per object category (as one would expect in natural scenes) is 0. Figure. 3 shows a histogram of non-zero counts across all object categories. Clearly, we see that the two datasets have a fair amount of count variability. As one might expect in everyday scenes, there is a bias towards the lower count values.

4.2 How do we evaluate counting?

Since we care about how close we are to the ground truth count, and not just how many times we predicted the right count, counting is similar to regression problems. We adopt the root mean squared error (RMSE) metric which is a standard metric for evaluating regression.

We first post-process the count predictions from each method by thresholding counts at 0, and rounding predictions to closest integers to get predictions $\hat{c}_i$. Given these predictions and ground truth counts c_i for a category c and image i, we compute the RMSE error as follows:

$$\text{RMSE}_c = \sqrt{\frac{\sum_{i=1}^{N}(\hat{c}_i - c_i)^2}{N}} \quad (2)$$

where N is the number of images in the dataset. We average the error across all categories to report a single number on the dataset (**mRMSE**).

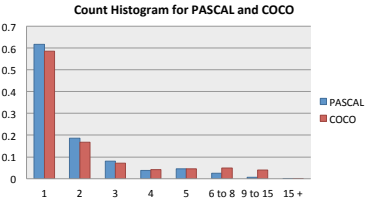


Fig. 3: Histogram of non-zero object counts (per image) in the PASCAL dataset (blue) and the MS COCO dataset (red). This plot is across all object categories.

We also evaluate mRMSE values for ground truth instances with non-zero counts (**mRMSE-nonzero**). This metric reflects more clearly how accurate the counts produced by a method (beyond predicting absence) are.

We also considered other choices for metrics, including a balanced error metric where error would be computed at each ground-truth count value for each category, and then averaged over all categories and count values. However, this would artificially alter the statistics of the dataset.

4.3 Methods and Baselines

We compute the following baselines for evaluating counting performance:

- `always-0`: predict most-frequent ground truth count on the VAL set (0)
- `mean`: predict the average ground truth count on the VAL set
- `always-1`: predict the most frequent non-zero value (1) for all classes
- `category-mean`: predict the average count *per* category on VAL
- `gt-class`: predict the *ground truth* classification labels as the counts. The idea is to check how much the actual count predictions make a difference beyond (perfect) image classification

We evaluate the following variants of counting approaches (see Sec. 3 for more details):

`detect`: We compare two methods for `detect`. The first method finds the best NMS and score thresholds as explained in Section. 3.1. The second method uses vanila Fast R-CNN as it comes out of the box, with the default NMS and score thresholds.

`glance`: We explore the following choices of features for `glance`: 1) vanilla classification features `noft`), 2) detection fine tuned features `ft`, and 3) `conv-3` features from a CNN trained for classification.

`aso-sub`: We examine three choices of grid sizes (Section. 3.1): 3×3, 5×5, and 7×7 and `noft` and `ft` features.

`ens`: Finally, we take the best methods from each category and perform counting by ensembling (`ens`) by averaging the count predictions of each method.

Approach	mRMSE	mRMSE-nonzero
`always-0`	0.65 ± 0.01	2.58 ± 0.06
`mean`	0.65 ± 0.01	2.48 ± 0.13
`always-1`	1.13 ± 0.01	1.97 ± 0.08
`category-mean`	0.63 ± 0.02	2.48 ± 0.13
`gt-class`	0.49 ± 0.03	1.89 ± 0.11
`detect`	0.51 ± 0.02	1.94 ± 0.05
`glance-noft-2L`	0.51 ± 0.02	1.83 ± 0.08
`aso-sub-ft-1L3` × 3	**0.43 ± 0.02**	**1.65 ± 0.09**
`ens`	**0.41 ± 0.01**	**1.69 ± 0.09**

Table 1: mRMSE and mRMSE-nonzero of various methods for counting on the PASCAL VOC 2007 TEST dataset. **Lower** is better.

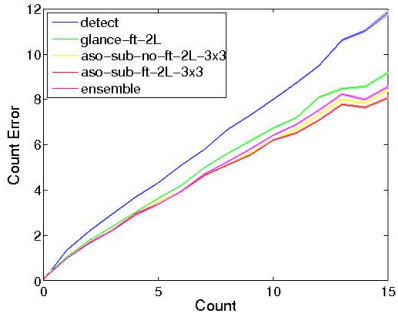
5 Results

We first describe our results on the PASCAL and MS COCO datasets. We then study how each of the methods perform for objects of various sizes and at different ground truth counts. We examine the performance of different versions of the `aso-sub` and `detect` models and provide qualitative insights. Further, we explore how counting can help object detection. Finally, we present proof-of-concept results on counting questions in the VQA [9] and COCO-QA [10] datasets. All the results presented in the paper are averaged on 10 random splits of the test set sampled with replacement.

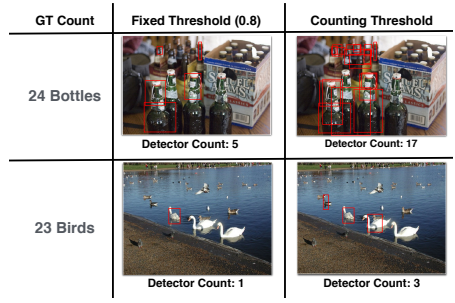
5.1 Counting Results

PASCAL VOC 2007: We first present results on PASCAL VOC 2007 (Table. 1) for the best performing variants (picked based on VAL) of each method. We see that `aso-sub` outperforms all other methods on mRMSE (0.43). Predicting all counts as zero (`always-0`) gives us an mRMSE of 0.65. Both `glance` and `detect` perform equally well as per mRMSE, while `glance` does slightly better on mRMSE-nonzero. An ensemble of all methods outperforms the best individual method (0.41 mRMSE). To put these numbers in perspective, we find that the difference of 0.04 mRMSE-nonzero between `aso-sub` and `ens` leads to a difference of 0.5% performance in our counting to improve detection application (Sec. 5.6). We also experiment with `conv3` features to regress to the counts, similar to Zhang. et.al. [24]. We find that `conv3` gets mRMSE of 0.63 which is much worse than `fc7`. We also tried PCA on the `conv3` features but that did not improve performance. This indicates that our counting task is indeed more high level and needs to deal with object semantics (unlike previous crowd/cell counting efforts).

MSCOCO: We next present results on MS COCO for the best performing `detect`, `glance` and `aso-sub` methods from PASCAL. The results are summarized in Table. 2. We find that `aso-sub` does the best on both mRMSE and



(a) Count vs Count Error



(b) Detector Count Visualization

Fig. 4: **(a)** We plot the count (x-axis) against the RMSE error (across all categories) with error bars (too small to be visible) at that count on the Count-Test split of the MS COCO dataset. We find that the `aso-sub-ft-1L3` $\times$ 3 approach performs really well at higher count values whereas the `ens` performs the best at lower count values (zoomed in). **(b)**We show the ground truth count (first col.), outputs of `detect` with a default threshold of 0.8 (second col.), and ouptuts of `detect` with hyperparameters tuned for counting (third col.). We can see that choosing a different threshold allows us to trade-off localization accuracy for counting accuracy (see bottle image). Thus, the method finds partial evidence for counts, even if it cannot localize the full object.

mRMSE-nonzero. A comparison indicates that the `always-0` baseline does a lot better on MSCOCO than on VOC 2007. This is because MSCOCO has many more categories than VOC 2007. Thus, the chances of any particular object being present in an image decrease compared to VOC 2007.

Approach	mRMSE	mRMSE-nonzero
`always-0`	0.53 $\pm$ 0.01	3.02 $\pm$ 0.02
`mean`	0.54 $\pm$ 0.00	2.48 $\pm$ 0.13
`always-1`	1.12 $\pm$ 0.00	2.39 $\pm$ 0.03
`category-mean`	0.52 $\pm$ 0.00	2.97 $\pm$ 0.01
`gt-class`	0.43 $\pm$ 0.00	2.36 $\pm$ 0.02
`detect`	0.5 $\pm$ 0.00	2.79 $\pm$ 0.00
`glance-ft-1L`	0.42 $\pm$ 0.00	2.24 $\pm$ 0.02
`aso-sub-ft-1L3` $\times$ 3	**0.39 $\pm$ 0.00**	**2.19 $\pm$ 0.02**
`ens`	**0.39 $\pm$ 0.00**	**2.19 $\pm$ 0.02**

Table 2: mRMSE and mRMSE-nonzero of various methods for counting on Count-Test set of the MS COCO [35] dataset. **Lower** is better.

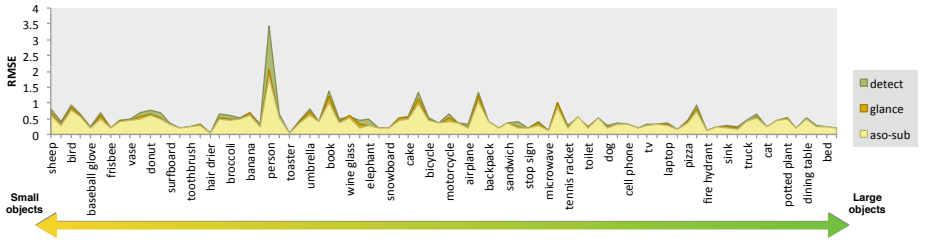


Fig. 5: We visualize RMSE error (y-axis) for `glance`, `detect`, and `aso-sub` across categories of various sizes on the MS COCO dataset. On the x-axis we order categories from smaller objects (left) to larger objects (right). We find that `aso-sub` performs consistently for a wide range of category sizes

5.2 Analysis of the predicted counts

We analyze the trends on MS COCO further to reveal some insights. We first look at the count vs. count error performance, and then characterize performance as a function of object size.

Count vs. count error: We analyze the performance of each of the methods at different count values on the MS COCO Count-Test set (Fig. 4a). In addition to the methods in Table. 1, we also evaluate `aso-sub-noft-1L3` $\times$ 3 since it is the second best performing method (0.44 mRMSE) after `aso-sub-ft-1L3` $\times$ 3, excluding `ens`.

We pick each count value on the x-axis (Fig. 4a) and compute the RMSE error over all the instances at that count value. Interestingly, we find that the two `aso-sub` approaches (with and without detection fine-tuning) work really well across a range of count values. This supports our intuition that `aso-sub` is able to capture partial counts (from larger objects) as well as integer counts (from smaller objects) really well. Of the two `aso-sub` approaches, `aso-sub-ft-1L3`$\times$3 method works better, likely because the detector fine-tuned features might be better suited to local image statistics than the *global* classification features. For lower count values, `ens` does the best (Fig. 4a zoomed in). However, for counts > 5 the `ens` performance starts tailing off. We hypothesize that this might be because the `detect` starts performing significantly worse at higher counts.

Object size vs. count error: We compare `glance`, `aso-sub`, and `detect` on their performance for object categories of various sizes on the MS COCO dataset (Fig. 5). To get the object size, we sum the number of pixels occupied by an object across images where the object occurs in the MS COCO VAL set and divide this number by the average number of (non-zero) instances of the object. This gives us an estimate of the expected size occupied per instance of an object. We show a sorting of smaller to larger categories on the x-axis in Fig. 5. We find that `aso-sub` does consistently well across the spectrum of object sizes from sheep and bird to pizza and sink. For larger object sizes, all the methods seem to perform competitively. This also indicates that `aso-sub` is able to capture

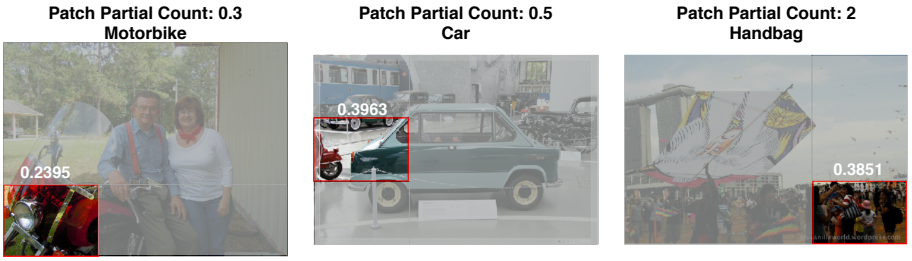


Fig. 6: We visualize fractional ground truth for three categories, along with predictions of the `aso-sub-ft-1L3` $\times 3$ model (in white) above each of the grid-cells. We find that the model performs fairly well on the fractional count-prediction tasks for the left and middle image, but fails to count as many handbags in the third image. Note how these gird-cells (eg. motorcycle image) can often capture part-like characteristics

partial ground truths well, since the counts for larger categories will necessarily be partial/fractional.

5.3 Counting by Detection (`detect`):

We tune the hyperparameters of Fast R-CNN in order to find the setting where the mRMS error is the lowest, on the Count-Val split of MSCOCO. We show some qualitative examples of the detection ground truth, the performance without tuning for counting (using black-box Fast R-CNN), and the performance after tuning for counting on the PASCAL dataset in (Figure. 4b). We use untuned Fast R-CNN at a score threshold of 0.8 and NMS threshold of 0.3, as used by Girshick *et.al.* [11] in their demo. At this configuration, it achieves an mRMSE of 0.52 on Count-Test split of MSCOCO. Thus, we achieve a gain of 0.02 mRMSE by tuning the hyperparameters for `detect`.

5.4 Counting by Associative Subitizing (`aso-sub`):

We next analyze how different design choices in `aso-sub` affect performance on PASCAL. We pick the best performing `aso-sub-ft-1L3` $\times$ 3 model and vary the grid sizes (as explained in Sec. 4). We experiment with 3×3, 5×5, and 7×7 grid sizes. We found that the performance of the 3×3 grid was the best (Figure. 7). Performance deteriorates significantly as we reach 7×7 grids. This indicates that there is indeed a sweet spot in the discretization as we interpolate between the `glance` and `detect` settings[1].

[1] Going from 1×1 to 3×3, one might argue that the gain in performance is due to more (augmented) training data. However, from the diminishing performance on increasing grid size to 5×5 (which has even more data to train from), we hypothesize that this is not the case.

We next examine qualitatively how well our model captures partial real-valued counts in grid-cells. We pick patches in MSCOCO Count-Test whose ground truth counts are in the range 0.3 ± 0.001, 0.5 ± 0.001 and 2 ± 0.001. We then visualize the predictions our `aso-sub` model on those patches and highlight success and failure cases (Figure. 6). Success cases are instances where the ratio of partial target count to the predicted count is between 0.5 to 2, while the rest are failure cases. All results are on 3×3 ground truth grids.

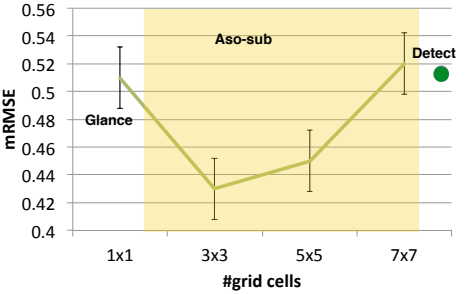


Fig. 7: Size of `aso-sub` grid (x-axis) vs. mRMSE (y-axis). `aso-sub` (shaded region) conceptually explores a continuum between `glance` and `detect` approaches. We find that there exists a sweet spot (3×3), where performance on counting is the best as we explore this continuum.

5.5 Ensemble of Counting Methods (`ens`):

We construct two ensembles and compare them. The first uses `detect`, `glance`, `aso-sub-ft-1L3` $\times$ 3 and `aso-sub-nofn-1L3` $\times$ 3. The second one is constructed using only `detect` and `glance`. We find that the first method gets mRMSE values of 0.41 ± 0.01 on PASCAL and 0.39 ± 0.00 on MS COCO. Using the second `ens` increases errors by 9.7% relative error on PASCAL and 10.2% on MS COCO. Thus, `aso-sub` is crucial to the performance of the ensemble. Overall, `ens` improves performance over individual methods on both PASCAL and MS COCO.

5.6 Counting to Help Detection

We now explore whether a counting can help *improve* detection performance (on the PASCAL dataset). Detectors are typically evaluated via the Average Precision (AP) metric, which involves a full sweep over the dynamic range of score-thresholds for the detector. While this is useful investigative tool, in any real application (say autonomous driving), the detector must make hard decisions at some fixed threshold. This threshold could be chosen on a per-image or per-category basis. Interestingly, if we knew *how many* objects of a category are present, we could simply set the threshold so that those many objects are detected. As a baseline, we do a sweep over the thresholds for each category to find the threshold that maximizes F-measure for that category. We call this the `base` detector. Note that since our goal is to intelligently pick a threshold for the detector, computing AP (which involves a sweep over the thresholds) is not possible. Hence, to quantify detection performance, we first assign to each detected box to one ground truth box with which it has the highest overlap. Then for each ground truth box, we check if any detection box has greater than 0.5 overlap. If so, we assign a match between the ground truth and detection, and take them out of the pool of detections and ground truths. Through this

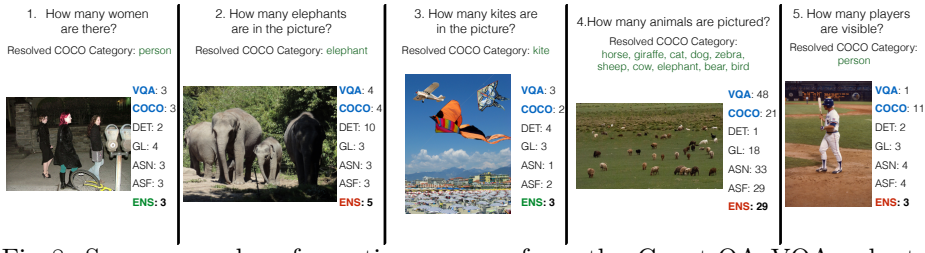


Fig. 8: Some examples of question-answers from the Count-QA VQA subset. Given a question, we parse the nouns and resolve correspondence to COCO categories. The resolved ground truth category is denoted after the question. We show the VQA ground truth and MS COCO dataset resolved ground truth counts, followed by outputs from `detect`(DET), `glance`(GL), `aso-sub-noft-1L3` $\times$ 3 (ASN), `aso-sub-ft-1L3` $\times$ 3 (ASF), and `ens`(ENS).

procedure, we obtain a set of true positive and false positive detection outputs. With these outputs we compute the precision and recall values for the detector. Finally, we compute the F-measure using these precision and recall values, and average the F-measure values across images and categories. We call this the **mF** (mean F-measure) metric. With a fixed per-category score threshold, the `base` detector gets a performance of 15.26% mF. With ground truth counts to select thresholds, we get a best-case `oracle` performance of 20.17%. Finally, we pick the outputs of `ens` model and use the counts to set the threshold. Our counting methods undercount more often than they overcount[2], a high count implies that the ground truth count is likely to be even higher. Thus, for counts of 0, we default to the `base` thresholds and for the other predicted counts, we use the counts to set the thresholds. With this procedure, we get a gain of 1.79% mF over the `base` performance.Thus, counting can be used as a complimentary signal to aid detector performance, by intelligently picking the detector threshold in an image specific manner.

5.7 VQA Experiment Setup

We explore how well our counting approaches do on simple counting questions. Recent work [9,10,36] has explored the problem of answering free-form natural language questions for Images. One of the recently released large-scale datasets in the space is the Visual Question Answering [9] dataset. For this dataset, human written captions are collected on Amazon Mechanical Turk (AMT), where workers are given the prompt to ask a question to *stump a smart robot*. We also evaluate using the COCO-QA [10] which automatically generates questions from human captions. Around 10.28% and 7.07% of the questions in VQA and COCO-QA are "how many" questions related to counting objects. Note that both the datasets use images from the MSCOCO [34] dataset. We apply our counting models, along with some basic natural language pre-processing to answer some of these questions.

[2] See appendix for more details.

Approach	RMSE(VQA)	RMSE(COCO-QA)
`detect`	3.05 ± 0.2	3.26 ± 0.24
`glance`	2.15 ± 0.09	1.95 ± 0.07
`aso-sub-ft-1L3` × 3	2.08 ± 0.06	1.69 ± 0.08
`aso-sub-noft-1L3` × 3	2.09 ± 0.09	1.69 ± 0.08
`ens`	**2.06 ± 0.09**	**1.65 ± 0.07**
SOTA VQA [37]	2.42 ± 0.35	N/A

Table 3: Performance of various methods on counting questions in the **Count-QA** splits of the VQA dataset and COCO-QA datasets respectively

Given the question "how many bottles are there in the fridge?" we need to reason about the object of interest (bottles), understand referring expressions (in the fridge) *etc.*Note that since these questions are free form, the category of interest might not exactly correspond to an MSCOCO category. We tackle this ambiguity by using word2vec embeddings [38]. Thus given a free form natural language question, we extract the noun from the question and compute the closest MSCOCO category by checking similarity of the noun with the categories in the word2vec embedding space. In case of multiple nouns, we just retain the first noun in the sentence (since how many questions typically have the subject noun first). We then run the counting method for the COCO category. More details can be found in the appendix. Note that parsing referring expressions is still an open research problem [39,40]. Thus, we filter questions based on an "oracle" for resolving referring expressions. This oracle is constructed by checking if the ground truth count of the COCO category we resolve using word2vec matches with the answer for the question. Evaluating only on these questions allows us to isolate errors due to inaccurate counts, on both datasets. We evaluate our outputs using the RMSE metric. We use this procedure to compile a list of 2084 + 2374 questions (**Count-QA**) from the VQA and COCO-QA datasets respectively, to evaluate on. We will publicly release our Count-QA subsets to help future work on counting.

We report performance separately on each dataset in Table. 3. We find that the `ens`, `aso-sub`, and `aso-sub-noft-1L3`×3 methods achieve similar performance, whereas the `glance` method does slightly worse on both datasets, and `detect` achieves the worst performance. We also evaluate a state-of-the-art VQA model [37] on the Count-QA VQA subset and find that `ens` does better.

6 Discussion

Our work opens up interesting avenues for future work to focus upon. For example, one could count with object completeness constraints in the `aso-sub` method. That is, we can use context that says that if 0.3 fraction of an object is in some box, an adjacent box should probably have the object as well. It would also be interesting to explore label based context for counting. That is, if we knew that four chairs are present, is a table also likely to be there?

7 Conclusion

We introduce the problem of counting *everyday* objects in *everyday* scenes. We propose three approaches to this problem: `detect`, `glance` and `aso-sub`. We find that our `aso-sub` method, which divides the image into grids and counts in each grid-cell using subitizing performs better than `detect` and `glance`. We evaluate the relative strengths, weaknesses and biases of our approaches. We then propose an ensemble of counting methods (`ens`) which improves counting performance. Further, we study how counting can be used to improve detection. Finally, we present proof-of-concept experiments on answering "how many?" questions in visual question answering tasks. Our code and datasets will be made publicly available.

References

1. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. Computer Vision and Pattern Recognition (CVPR) **1** (2001) I—-511—-I—-518
2. Felzenszwalb, P., McAllester, D., Ramanan, D.: A discriminatively trained, multiscale, deformable part model. In: 26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR. (2008)
3. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. (11 2013)
4. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. (jun 2015)
5. Carreira, J., Caseiro, R., Batista, J., Sminchisescu, C.: Semantic segmentation with second-order pooling. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Volume 7578 LNCS. (2012) 430–443
6. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 3431–3440
7. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. (2012) 1097–1105
8. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3) (2015) 211–252
9. Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C.L., Parikh, D.: VQA: visual question answering. CoRR **abs/1505.00468** (2015)
10. Ren, M., Kiros, R., Zemel, R.: Exploring models and data for image question answering. (5 2015) 11
11. Girshick, R.: Fast r-cnn. (4 2015)
12. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. (9 2014)
13. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. (10 2013)

14. Clements, D.H.: Subitizing: What is it? why teach it? Teaching children mathematics **5**(7) (1999) 400
15. Klein, A., Starkey, P.: Universals in the development of early arithmetic cognition. New Directions for Child and Adolescent Development **1988**(41) (1988) 5–26
16. Kaufman, E.L., Lord, M.W., Reese, T.W., Volkmann, J.: The discrimination of visual number. The American Journal of Psychology **62**(4) (1949) 498–525
17. Freeman, F.N.: Grouped objects as a concrete basis for the number idea. The Elementary School Teacher **12**(7) (1912) 306–314
18. Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T., Smeulders, A.W.M.: Selective search for object recognition. International Journal of Computer Vision **104**(2) (2013) 154171
19. Cheng, M.M., Zhang, Z., Lin, W.Y., Torr, P.H.S.: BING: Binarized normed gradients for objectness estimation at 300fps. In: IEEE CVPR. (2014)
20. Zitnick, C.L., Dollár, P.: Edge boxes: Locating object proposals from edges. In: ECCV, European Conference on Computer Vision (September 2014)
21. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. (6 2015)
22. Gidaris, S., Komodakis, N.: Object detection via a multi-region & semantic segmentation-aware cnn model. (5 2015)
23. Wan, L., Eigen, D., Fergus, R.: End-to-end integration of a convolution network, deformable parts model and non-maximum suppression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 851–859
24. Zhang, C., Li, H., Wang, X., Yang, X.: Cross-Scene Crowd Counting via Deep Convolutional Neural Networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 833–841
25. Seguí, S., Pujol, O., Vitrià, J.: Learning to count with deep object features. (may 2015)
26. Chan, A.B., Vasconcelos, N.: Bayesian poisson regression for crowd counting. In: 2009 IEEE 12th International Conference on Computer Vision, IEEE (9 2009) 545–551
27. Lempitsky, V., Zisserman, A.: Learning To Count Objects in Images. In: Advances in Neural Information Processing Systems. (2010) 1324–1332
28. Chan, A.B., Vasconcelos, N.: Privacy preserving crowd monitoring: Counting people without people models or tracking. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, IEEE (6 2008) 1–7
29. Zhang, J., Ma, S., Sameki, M., Sclaroff, S., Betke, M., Lin, Z., Shen, X., Price, B., Měch, R.: Salient object subitizing. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2015)
30. Hinton, G.E., McClelland, J.L., Rumelhart, D.E.: Distributed representations. Parallel Distributed Processing (1986) 77–109
31. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv (2015)
32. : Rmsprop. `http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`
33. Everingham, M., Eslami, S.M.A., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision **111**(1) (January 2015) 98–136
34. Lin, T., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV. (2014)
35. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV. (2014)

36. Malinowski, M., Rohrbach, M., Fritz, M.: Ask your neurons: A neural-based approach to answering questions about images. (5 2015)
37. Jiasen Lu, Xiao Lin, D.B., Parikh, D.: Deeper lstm and normalized cnn visual question answering model. `https://github.com/VT-vision-lab/VQA_LSTM_CNN` (2015)
38. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems. (2013) 3111–3119
39. Kazemzadeh, S., Ordonez, V., Matten, M., Berg, T.L.: Referit game: Referring to objects in photographs of natural scenes. In: EMNLP. (2014)
40. Sadovnik, A., Gallagher, A.C., Chen, T.: It's not polite to point: Describing people with uncertain attributes. In: CVPR, IEEE (2013) 3089–3096
41. Loper, E., Bird, S.: Nltk: The natural language toolkit. In: In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics. (2002)

Appendix

We organize the appendix as follows.

I. We first discuss the architectures we tested for the `glance` and `aso-sub` methods (Sec. 3.2 in main paper) in more detail with performance comparisons.
II. We provide some analysis of the counting methods in terms of whether they tend to undercount or overcount.
III. We then provide some more details on the VQA experiment.

I: Performance comparisons

We explain details of the architectures for `glance` and `aso-sub` in Sec. 3.2 in the main paper. We provide more details here. For both the `glance` and `aso-sub` methods, we search over the following space of architectures for the MLP: 1) we first sweep over the hidden layer sizes from $[250, 500, 1000, 1500, 2000]$, 2) we then vary the number of hidden layers between 1 and 2 (with the selected hidden layer size from step 1). For these settings, for each method `glance` and `aso-sub` we search for the best performing architecture for both `ft` (detection finetuned fc7) as well as `noft` (classification features). For the `aso-sub` method, we find the best performing architecture across different grid sizes (3×3, 5×5, 7×7) and pick the best performing architecture as the `aso-sub` architecture (and evaluate all grid sizes with that architecture). We first narrow down to some design choices with 3×3 and then compare different grid sizes.

Below we give the exact details of how performance changes on PASCAL VOC 2007 with different choices of architectures.

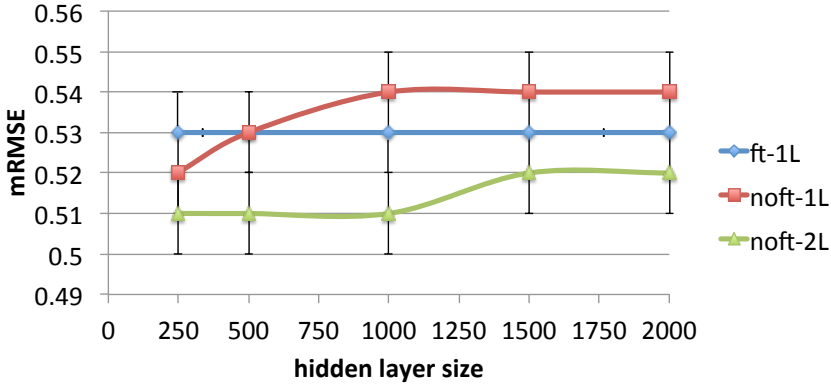


Fig. 9: Performance variations (mRMSE) on PASCAL VOC 2007 VAL set with different sizes of hidden layers (x-axis) for `glance`. We show performance with 1 hidden layer for `ft` (detection finetuned features) as well as `noft` (classification features). We observe that the classification features do better. We then increase the number of hidden layers to 2 for the model `noft` features and find that it does the best

`glance`

We find that the performance for `ft-1L` remains more or less constant as we change the size of the hidden layers (Fig. 9). In contrast, the `noft-1L` model does best at smaller hidden layer sizes. A two hidden layer `noft` model does better than both `1L` models. Intuitively, this makes sense since the `noft` features are better suited to global image statistics than the detection finetuned `ft` features.

`aso-sub` $\mathbf{3 \times 3}$

We next contrast different design choices for the `aso-sub` 3×3 method. Details of how the performance changes with different grid sizes in `aso-sub` are provided in Sec. 5.4 in the main paper. In particular, just like the previous section, we study the impact of hidden layer sizes and number of hidden layers, as well as the choice of features (`ft` vs `noft`) for the `aso-sub` 3×3 model (Fig. 10). We find that the detection finetuned `ft` features do much better than the classification features `noft` for `aso-sub`. This is likely because the `ft` features are better adapted to statistics of local image regions than the `noft` image classification features. We also find that increasing the number of hidden layers does not improve performance over using a single hidden layer, unlike `glance`.

`aso-sub`

We next compare how the performance of `aso-sub` varies as we change the size of the grids on the PASCAL VOC 2007 VAL set. We pick the best performing `aso-sub` 3×3 features (`ft`) and number of hidden layers (1 hidden layer). We

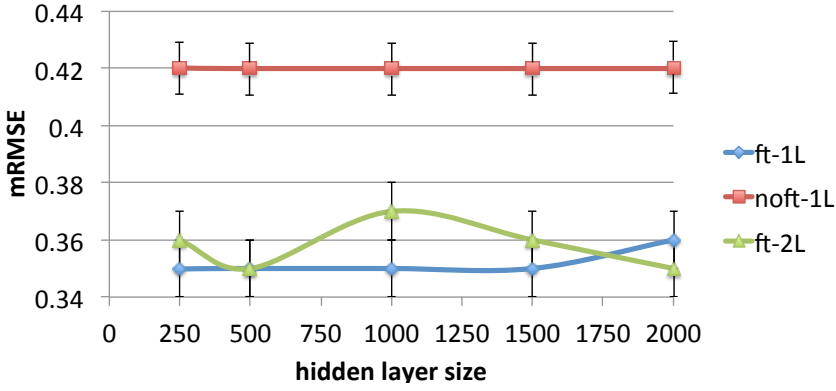


Fig. 10: Performance variations (mRMSE) on PASCAL VOC 2007 VAL set with differentsizes of hidden layers (x-axis) for `aso-sub` 3×3. We show performance with 1 hidden layer for `ft` (detection finetuned features) as well as `noft` (classification features). We observe that the detection finetuned features do better. We then increase the number of hidden layers to 2 for the model `ft` features and find that it does not give a substantial performance boost over `ft`

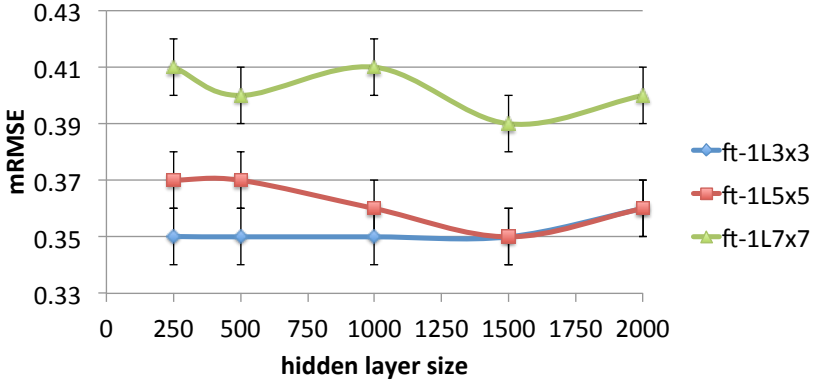


Fig. 11: Performance variations (mRMSE) on PASCAL VOC 2007 VAL set with different sizes of hidden layers (x-axis) for `aso-sub`. We compare the 3×3, 5×5, and 7×7 models, and find that the 3×3 model performs the best

then vary the size of the hidden layer and compare the performance of 3×3, 5×5, and 7×7 `aso-sub` approaches (Fig. 11). We find that 3×3 and 5×5 models do much better than the 7×7 model. The performance of 3×3 is slightly better than the 5×5 model. A similar comparison on the TEST set can be found in Sec. 5.4 of the main paper.

II: Overcounting vs. Undercounting

We study whether the outputs given by the approaches undercount, or overcount. We first filter out all instances where the ground truth count is 0 (since we cannot undercount 0). We then check if the predicted count is greater than the ground truth (overcounting) or lesser (undercounting). We perform this analysis on COCO Count-Test set (Section. 4.1 in main paper). We find that all three methods (`detect`, `glance`, and `aso-sub`) have a tendency to undercount rather than overcount. The detect method undercounts the most since it is finding evidence for the complete object (as opposed to parts) and it is difficult to detect objects at smaller scales. Indeed, we find that the detector undercounts 86% of the time. The `aso-sub` method undercounts 68% of the time. The `glance` method undercounts the least (62%), presumably because it optimizes for the full counting objective. As a result it learns to overcome the bias of lower counts being more common better than the other methods (`detect` and `aso-sub`).

III: VQA experiment

We next elaborate on more details of the VQA experiments described in Sec. 5.7 in the main paper. More specifically we discuss how we pre-process ground truth to make it numeric and give details of how we solve correspondence between a noun in a question to counts of coco categories.

As reported in the paper, we use the VQA dataset [9] and COCO-QA [10] datasets for our counting experiments. We extract the "how many" type questions, which have numbers as answers. This includes both integers (VQA) and numbers written in text form (COCO-QA). We parse the latter into corresponding numbers on the COCO-QA dataset. That is "five" is mapped to "5". From the selected questions, we extract the Nouns (singular, plural, and proper), and convert them to their singular form using the Stanford Natural Language Parser (NLTK) [41].

We train word2vec word embeddings on Wikipedia[3] and use cosine similarities in the embedding space as word similarity. Using these we find the COCO category or COCO super-category that matches the most with the extracted nouns. These super-category annotations are available as part of the COCO dataset. We run our counting method for the COCO category, and consider it the answer. For the extracted nouns, if the best match is with a COCO super category, we sum the counts obtained by our counting methods for each of the COCO sub-categories belonging to the particular super-category. For example,

[3] https://www.wikipedia.org/

if the resolved noun is “animal, we sum the counts for “horse”, “giraffe”, “cat”, “dog”, “zebra”, “sheep”, “cow”, “elephant”, “bear”, and “bird” and use the output as our predicted count.