Learning to Count Objects in Natural Images for Visual Question Answering

https://github.com/Cyanogenoid/vqa-counting

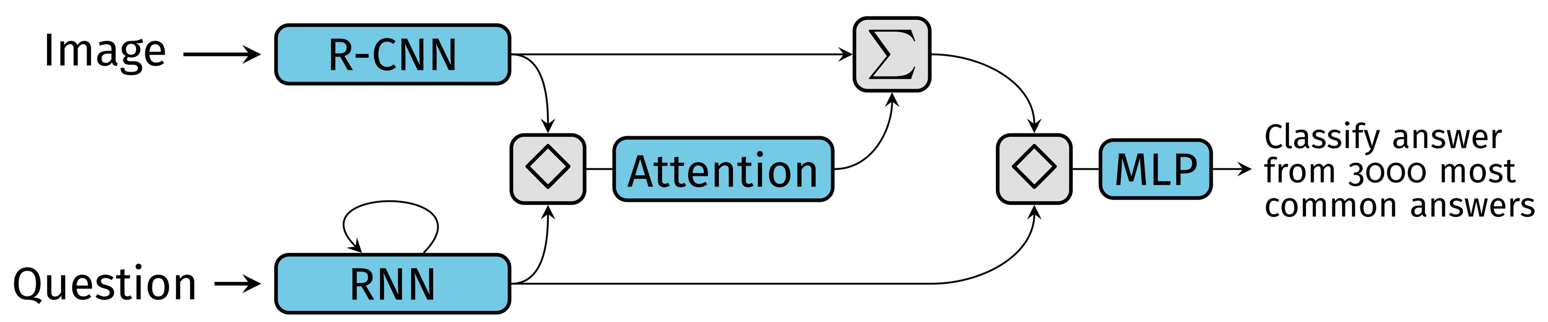
Yan Zhang, Jonathan Hare, Adam Prügel-Bennett

University of Southampton

Introduction

- **Summary**: Enabling VQA models to count by handling overlapping object proposals.
- Visual Question Answering (VQA): answer questions about an image.
- VQA is like a visual Turing test: natural images, human-posed questions, expects natural language answers.
- Counting questions ("how many ...?") are among easiest tasks in VQA for humans, but VQA models only fit to dataset biases so far.
- Contribution: Fully differentiable component that produces a deduplicated count, usable with any VQA model that uses soft attention.

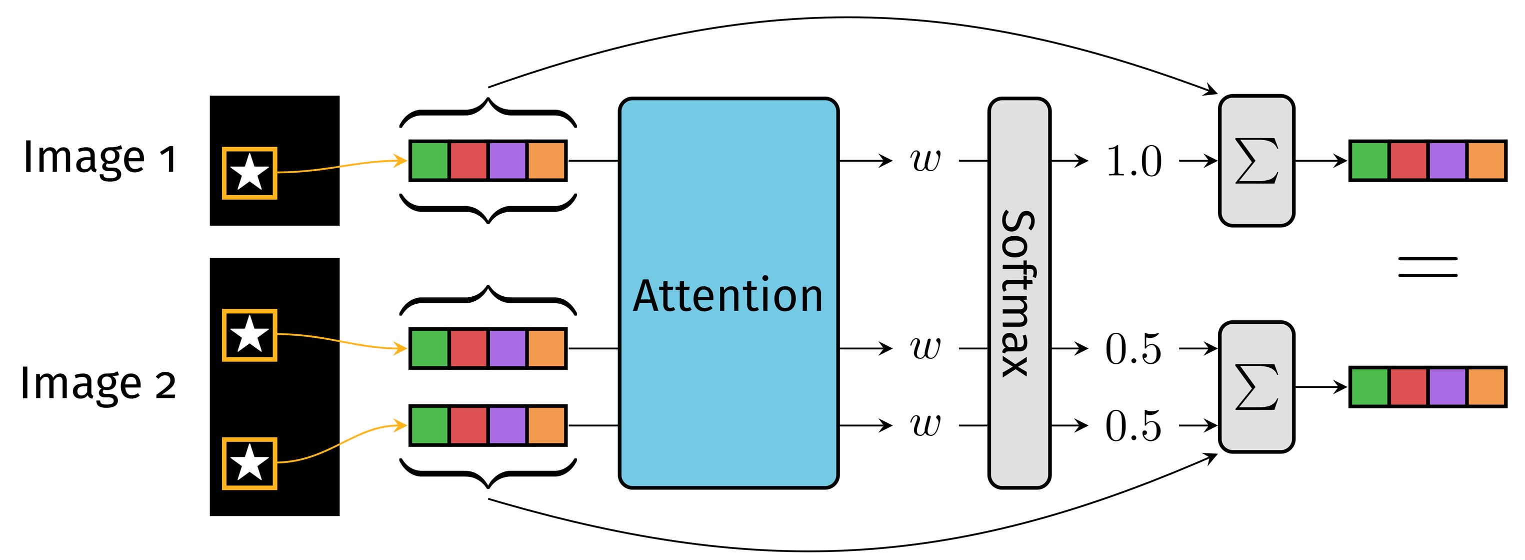
Existing VQA architectures



- ◇ stands for multimodal fusion: concatenate, add, multiply, bilinear, etc.
- $\sum$ takes a set of feature vectors and one or more corresponding attention maps to produce a single feature vector.

Problems

- Only weak supervision in form of noisy ground-truth answers.
- Complex scenes, occlusion of objects, inaccurate object proposals.
- Questions can be arbitrarily precise.
- **Major issue:** soft attention, which treats its input as set.
  - Issues when multiple objects of same type present, which breaks counting:



- Softmax normalises attention weights to sum to one.
- Resulting feature vector is exactly the same between the two images, all information about a possible count is lost.
- Changing softmax to sigmoid or using multiple attention glimpses does not help.

Method

Goal: Produce a count from attention map that:

- handles overlapping object proposals to avoid double-counting,
- is differentiable so we can backprop through it.

Key idea: treat object proposals as nodes in a graph, scale edge weights such that an accurate count can be recovered through the sum over edge weights.

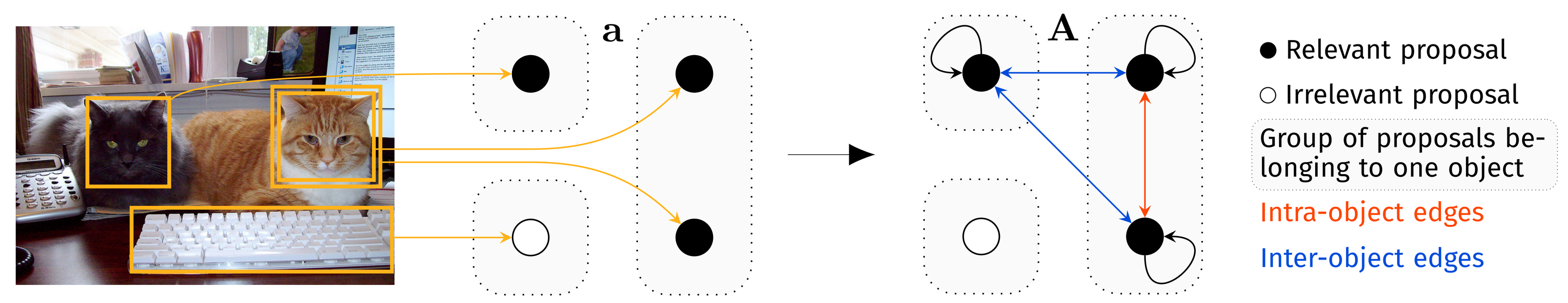
- Correct deduplication behaviour for extreme, *dataset-independent* cases[1] enforced through architecture. **Learned interpolation**[2] of correct behaviour for realistic cases.
  → only need to think about getting the extreme cases correct in the architecture, handling of partially overlapping proposals comes for "free".
- Edges are scaled such that the graph is **equivalent under a sum** to a graph without duplicates. Scaling is differentiable, unlike trying to delete duplicate nodes.

[1] Attention weights are either exactly 0 or 1 (not relevant or relevant proposal) and any pair of proposals is either fully distinct or fully overlapping (IoU of 0 or 1).

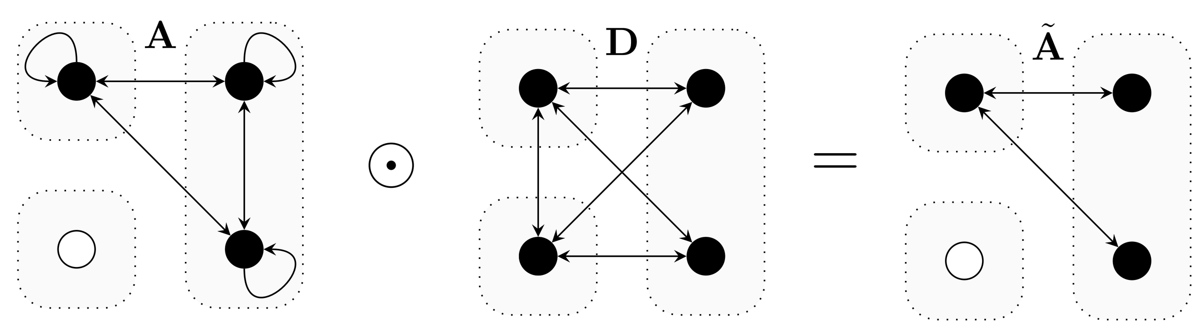
[2] This is achieved with individually parametrised piecewise linear functions f that have domain and range $[0, 1]$, are monotonic, and satisfy $f(0) = 0$, $f(1) = 1$.

Details

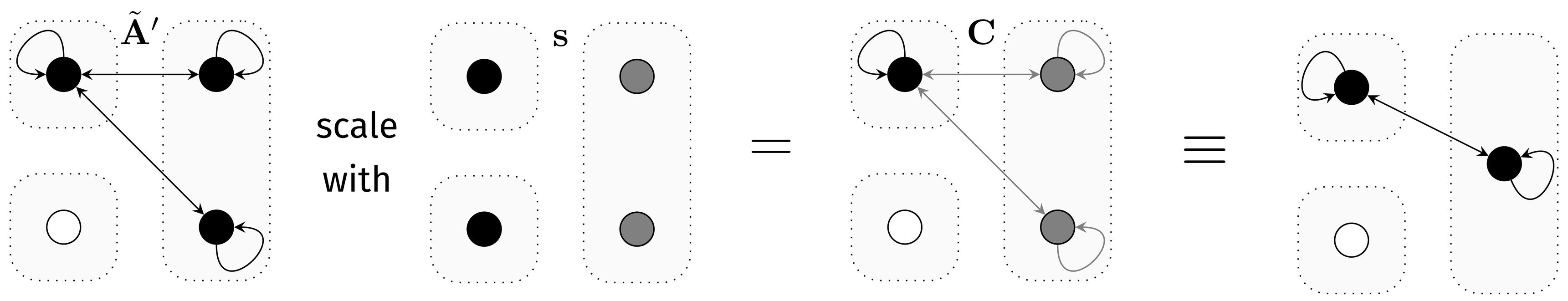
1. Expand vector of attention weights $\mathbf{a}$ to $\mathbf{A}$ using the outer product $\mathbf{a}\mathbf{a}^\mathsf{T}$. Interpret as adjacency matrix.



2. **Intra-object deduplication**: Mask away edges between overlapping proposals of $\mathbf{A}$ by multiplying with distance matrix $\mathbf{D}$ to obtain $\tilde{\mathbf{A}}$. Two nodes in $\mathbf{D}$ are connected if their corresponding proposals do not overlap.



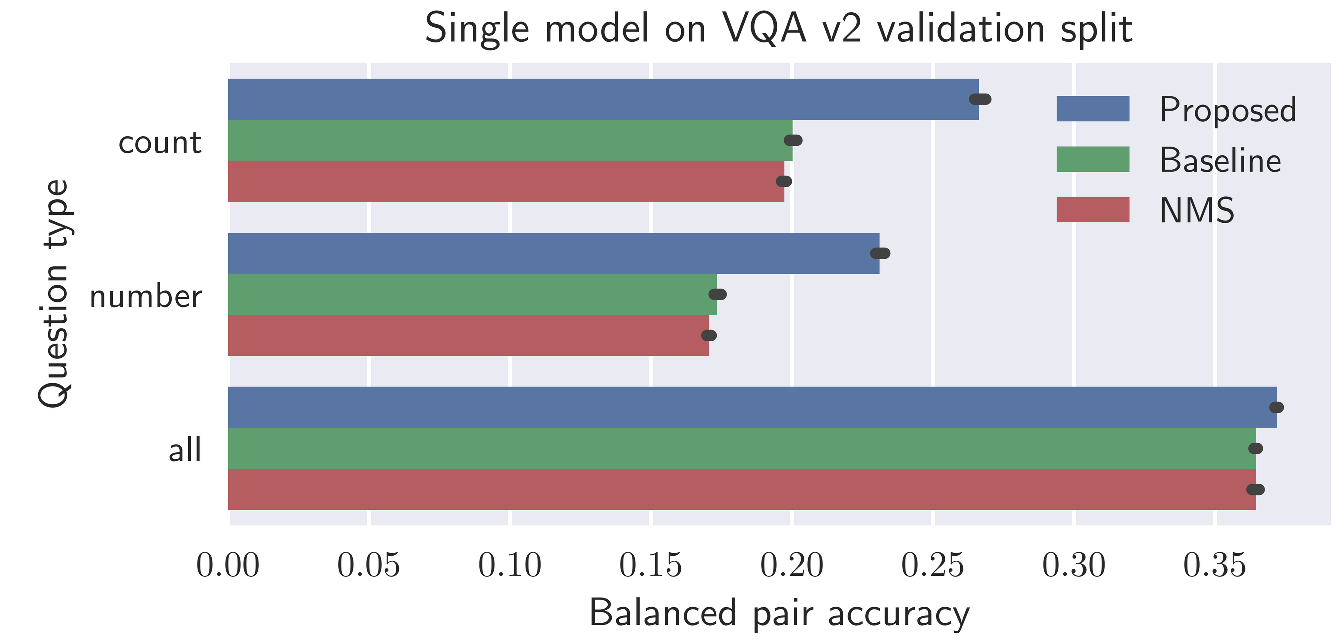
3. **Inter-object deduplication**: Compute similarity vector $\mathbf{s}$ from $\tilde{\mathbf{A}}$ that measures for each node the number of nodes with a similar neighbourhood. Add self-loops back to $\tilde{\mathbf{A}}$ to obtain $\tilde{\mathbf{A}}'$ and scale with $\mathbf{s}$ to obtain the final count matrix $\mathbf{C}$. Under a sum, $\mathbf{C}$ is equivalent to a graph that has no more than proposal per object.

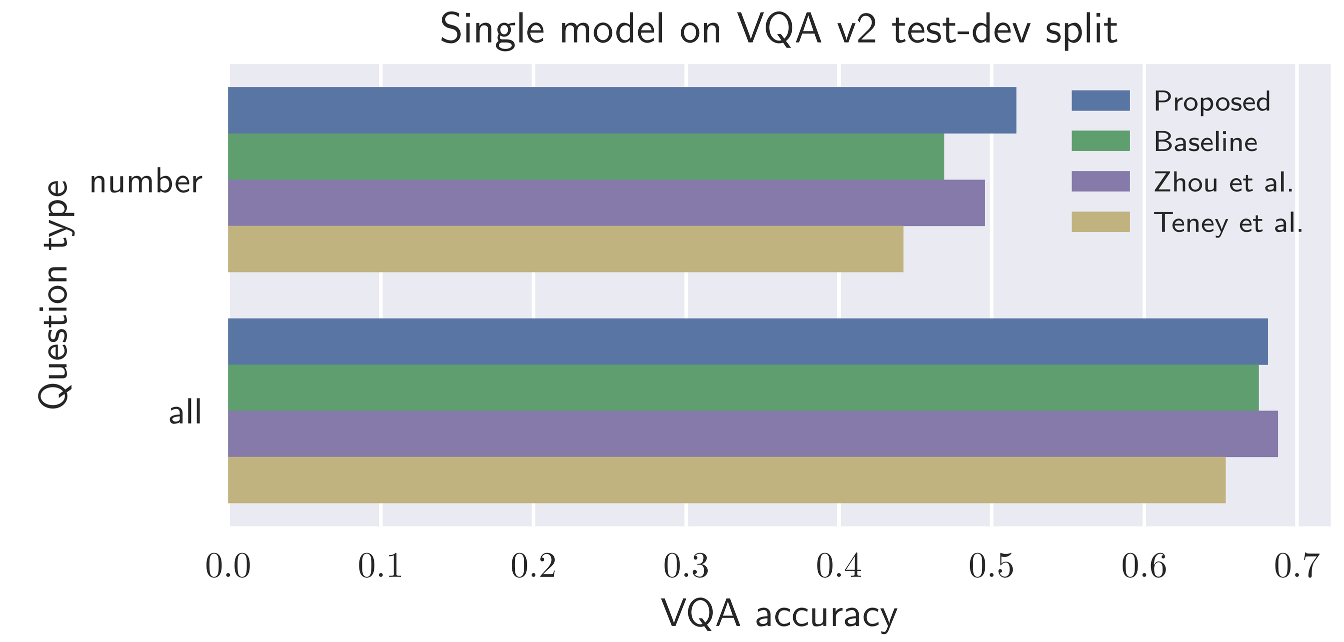


4. The final count c is $\sqrt{\sum_{i,j} C_{ij}}$. The square root undoes the squaring from taking the outer product (note: $(\sum_i a_i)^2 = \sum_{i,j} A_{ij}$). The count is one-hot encoded[3] and scaled with a learned confidence about the prediction. The resulting feature vector is fed into the answer classifier.

[3] Since c is a real number, a linear interpolation of the two one-hot vectors obtained when rounding c up and down is used to keep it differentiable.

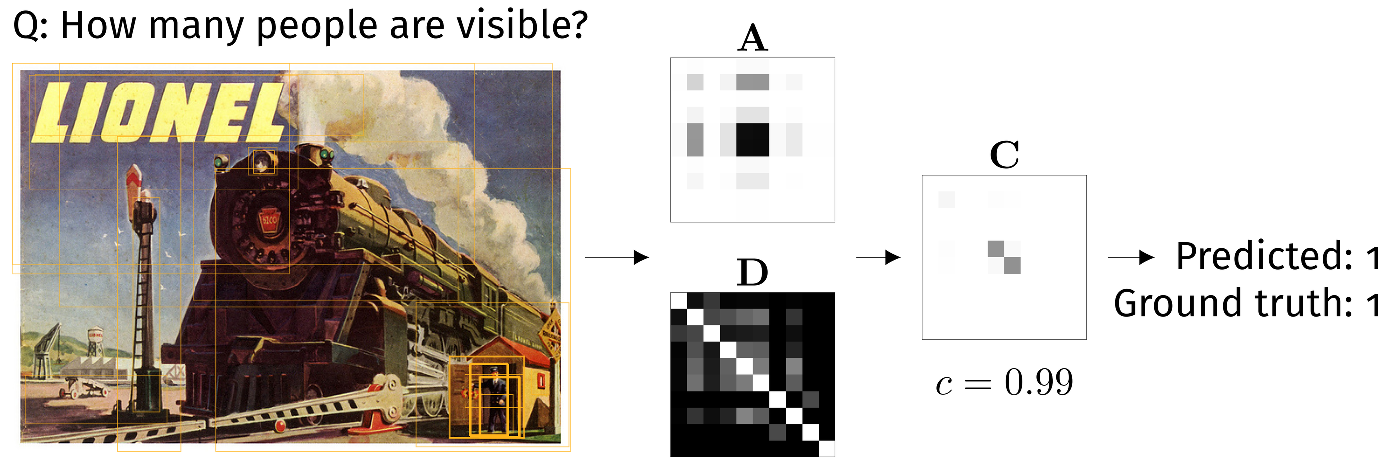
Results



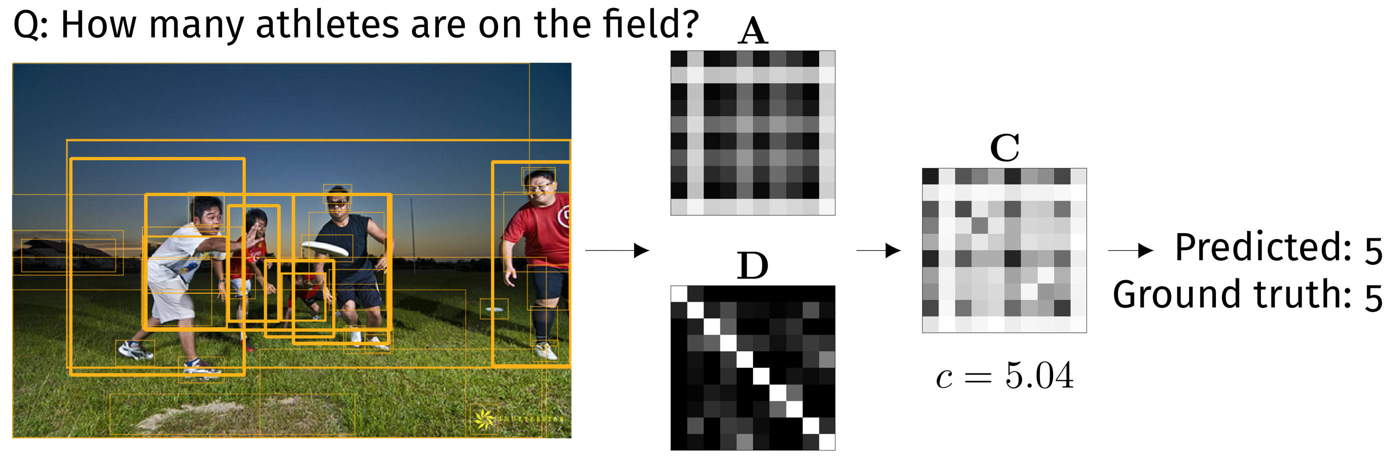


Examples

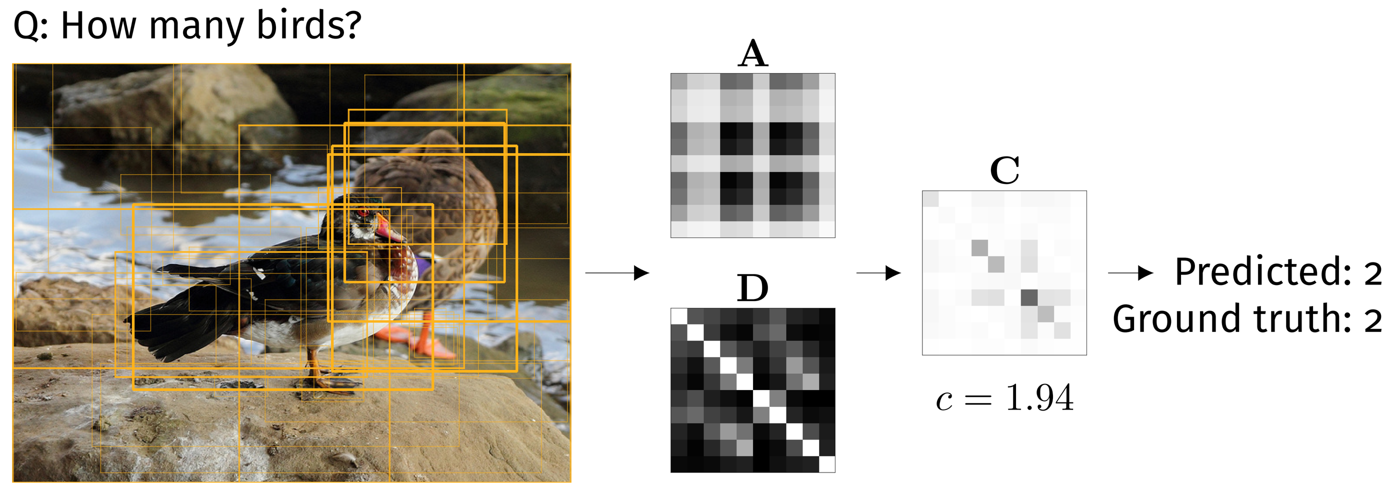
Q: How many people are visible?



Q: How many athletes are on the field?



Q: How many birds?



ICLR 2018