Pixels to Graphs by Associative Embedding (2017)

Alejandro Newell, Jia Deng

Resume

April 30, 2019

1 Introduction

In this paper the authors propose an end to end method for detection objects and their relationships in the form of graphs based on an associative embedding loss, the network in trained to define a complete graph from raw input image, the proposed supervision allows a network to better account for the full image context while making prediction so that the network reasons jointly over the entire scene graph rather than focusing on pairs of objects in isolation, the output is a get of triplets (object relation object), the objects are defined as a set of graph vertices and the relationshipts are the edges connecting the source and the destination objects.

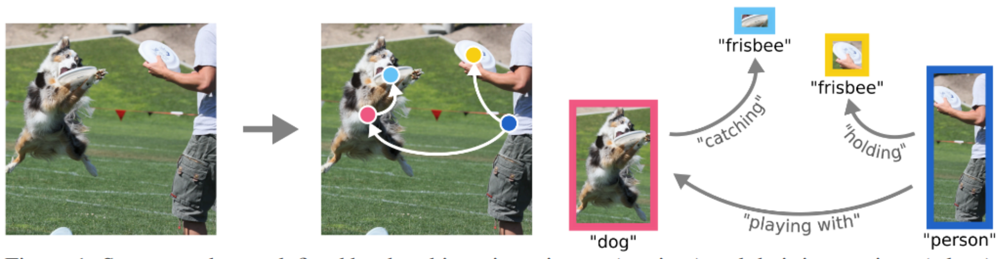

Figure 1: Scene graphs are defined by the objects in an image (vertices) and their interactions (edges). The ability to express information about the connections between objects make scene graphs a useful representation for many computer vision tasks including captioning and visual question answering.

2 Method

The objective is contruct a graph grounded in the space of the image pixels, in addition to identifying vertices of the graph we want to know the precise location of the objects of interest, and also the relationship between these objects that is captured by the edges of the graph.

More formally we consider a directed graph $G = (V, E)$. A given vertex $v_i \in V$ is grounded at a location (x_i, y_i) and defined by its class and bounding box. Each edge $e \in E$ takes the forme $i = (v_s, v_t, r_i)$ defining a relationship of type r_i from v_s to v_t. We train a network to explicitly define V and E. This training is done end-to-end on a single network, allowing the network to reasonfully over the image and all possible components of the graph when making its predictions.

The processing is end to end, but we might think of it two steps, first detect the graphs elements which are the objects and the relationships, and then connect them together.

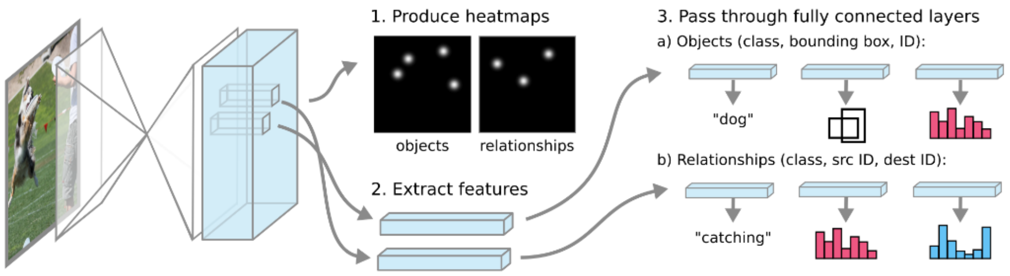
1. Produce heatmaps
objects
relationships
2. Extract features
3. Pass through fully connected layers
a) Objects (class, bounding box, ID):
"dog"
b) Relationships (class, src ID, dest ID):
"catching"

Figure 2: Full pipeline for object and relationship detection. A network is trained to produce two heatmaps that activate at the predicted locations of objects and relationships. Feature vectors are extracted from the pixel locations of top activations and fed through fully connected networks to predict object and relationship properties. Embeddings produced at this step serve as IDs allowing detections to refer to each other.

First step the network finds all of the vertices and edges that make up a graph. Each graph element is grounded at a pixel location which the network must identify, of the objects / vertices, the center of the object bounding box will serve as the grounding location, and the relationships / edges are grounded at the midpoint of the source and target vertices: $\left(\lfloor\frac{x_s+x_t}{2}\rfloor, \lfloor\frac{y_s+y_t}{2}\rfloor\right)$. With this grounding in mind, we can detect individual elements by using a network that produces per-pixel features at a high output resolution. The feature vector at a pixel determines if an edge or vertex is present at that location, and if so is used to predict the properties of that element. A convolutional neural network (Hour glass network) is used to process the image and produce a feature tensor of size $h \times w \times f = 64x64x256$. All information necessary to define a vertex or edge is thus encoded at particular pixel in a feature vector of length $f = 256$, we then take these feature maps and use a conv 1x1 with a signmoid to produce two heat maps, one for detection the vertices and one for the edges each element indicates the likelihood that a edge / vertex presence in a given image location, after applying a binary cross entropy and a threshold, the remaining detections are the condidates to be processed in the second step (the output size is important, the smaller the output size the more likely we'll endup with different objects / relation grounded at the same x, y location, so we must choose a good compromise between the computation and number of overlaps, the authors use a 64x64 output with a 512x512 input).

Second step is to connect the detected element using the associative embeddings losses, we take the f features at each detected location in the output feature maps, for the object instances we pass them a fully connected layer producing the class of the box, the regression of the anchors, and a 8-dimensionnal embedding of the object, for the relations, we pass them through a different FC layer producing the class / type of the relation, and two 8-d embeddings, one for the source and one for the target, and these embedding must be similar to the embeddings of the objects, and this is enforced using associative losses, for a given object / vertex $h_i \in R^8$, we find its relations / edges using the ground truth h'_{ik} (k from 1 ... K number of relations of a given vertex), and then get the predicted embedding of this edge, and the embeddings of this edge a the vertex must be similar:

$$L_{pull} = \frac{1}{\sum_{i=1}^{n} K_i} \sum_{i=1}^{n} \sum_{k=1}^{K_i} (h_i - h'_{ik})^2$$

And in the second term of the associative loss, we must push ways the embeddings of the objects, if two objects have a similar embeddings, L_{push} will be equal m and will converge twords zero after the embeddings becom different from each other, in the authors imlementation, $m = 8$:

$$L_{push} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \max\left(0, m - \|h_i - h_j\|\right)$$

Supporting overlapping detections To support the detection of different objects and relations in the grounded location, the authors define a number of slots for the objects / relation, $s_o = 3$ and $s_r = 6$ which are the slots availble for object and reltionhips respectively, so we can detect up to 3 objects location and 6 relations in the same, now we take the 256 vector for a given location, and pass them trough the feature extractor resulting in 3 x (class, bb regression, embeddings) for object and 6 x (type, source emb, target emb), but how do we find the matches between the ground truth and one of these slots, of the objecs first we concatenate the one hot encoding of the class and the box regression, and use the hangarian algorithm to find the best match from the 3 output slots for each ground truth and calculate the classification loss (CE) and regression (say MSE), for the relations, in a similar manner we concatenate the one hot encoding of the relation type, and the embeddings of the source and target objects (these embeddings are taken from the matched object slots), find the best match and calculate the associative losses.

In trainig, they only take a sample for the negatives, so that we don't penelise the correct relations that are now present in the ground truth, during inference, we take the set of objects and relation detected using the two head maps (after thresholding), and by comparing the source and target emb of the relations and the emb of the objects we can detect the correct triplets of obj relation obj.

3 Experiments

- SGGen: Detect and classify all objects and determine the relationships between them.
- SGCls: Ground truth object boxes are provided, classify them and determine their relationships.
- PredCls: Boxes and classes are provided for all objects, predict their relationships.

	SGGen (no RPN)		SGGen (w/ RPN)		SGCls		PredCls	
	R@50	R@100	R@50	R@100	R@50	R@100	R@50	R@100
Lu et al. [18]	–	–	0.3	0.5	11.8	14.1	27.9	35.0
Xu et al. [26]	–	–	3.4	4.2	21.7	24.4	44.8	53.0
Our model	6.7	7.8	9.7	11.3	26.5	30.0	68.0	75.2
Our model (03/2018)	**15.5**	**18.8**	–	–	**35.7**	**38.4**	**82.0**	**86.4**

Table 1: Results on Visual Genome. Updated numbers *(bottom row)* are the result of longer training with more efficient code. Full code and pretrained models are available at `https://github.com/umich-vl/px2graph`.

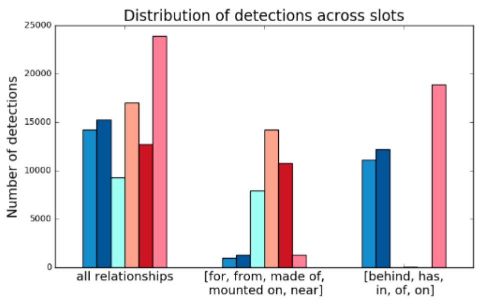

Figure 4: How detections are distributed across the six available slots for relationships.

Predicate	R@100	Predicate	R@100
wearing	87.3	to	5.5
has	80.4	and	5.4
on	79.3	playing	3.8
wears	77.1	made of	3.2
of	76.1	painted on	2.5
riding	74.1	between	2.3
holding	66.9	against	1.6
in	61.6	flying in	0.0
sitting on	58.4	growing on	0.0
carrying	56.1	from	0.0

Table 2: Performance per relationship predicate (top ten on left, bottom ten on right)