Graph Attention Network for contextual object detection

Marko Njegomir, Dušan Erdeljan
Soft computing 2021/2022, Teaching Assistant: Dragan Vidaković, Professor: dr Jelena Slivka

Problem

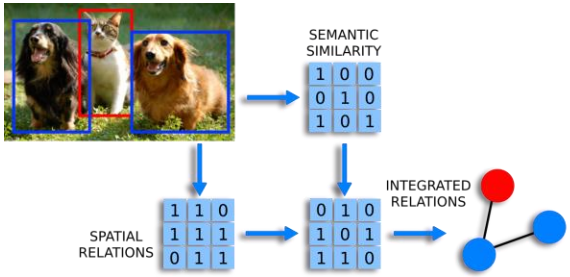
Most object detection algorithms consider each region proposal separately, without taking global image context into consideration. Out intuition is that some objects appear in the scenes alongside with some specific objects, or the objects appear in groups. We aim to solve this problem by using graphs to model semantic and spatial relations between objects in the image so that Graph neural network can improve classification.

Graph creation

Input to this module are region proposal bounding boxes and embeddings. We than consider each embedding to be a node in the graph and use region proposals (bounding boxes) to compute semantic similarity and spatial relations.

Semantic similarity between proposals is computed as cosine similarity.

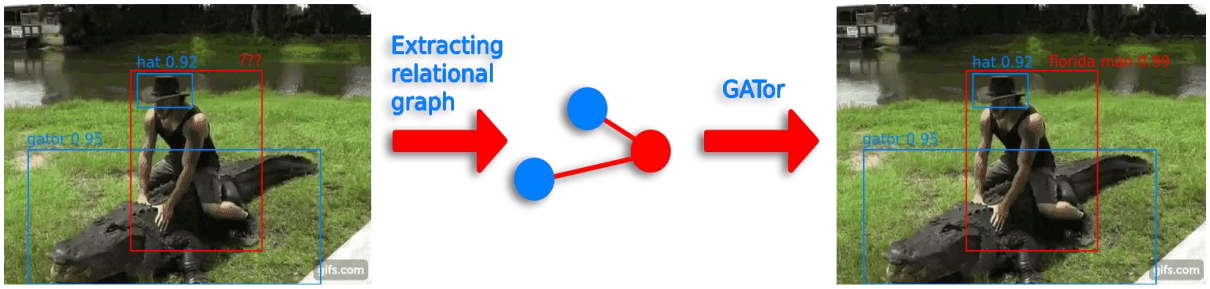
Spatial relations are modeled so that the strength of the relation is decreasing exponentialy based on the pixel distance between bounding boxes' centers. When creating spatial relations matrix we only consider K closest neighbours which are some distance aparat from the current object.



After we generate cosine similarity and spatial relations matrices, we than combine them into a single adjacency matrix by performing addition. Spatial relations matrix and cosine similarity matrices are filtered so that we remove small values (i.e. unlikely relations in the resulting graph and to remove self edges). Resulting combined matrix is again filtered to remove edges which corresponds to the objects with high bounding box overlap (IoU), since those proposals will be filtered out by the NMS algorithm in the end anyway.

Graph Attention Network

We create Graph Neutral Network with the stack of graph attention convolutional layers. Module recieves created graph and has a task to modify the input emebeddings of the nodes (which are outputed by FasterRCNN backbone), by taking contextual information into account which is embedded in the graph structure. Attention mechanism should learn to pay attention to some neighbouring nodes more than the others.



After the graph has been processed by GAT layer, the modified embeddings are then fed into classifier and bounding box regressor to make the final prediction. To improve the expressiveness of the attention function, we used GATv2 layer, which applies attention mechanism after processing the central node and its neighbouring nodes embeddings. Our GATor module can be applied to any backbone which produces region proposals with embeddings and can be used with any classifier and bounding box regressor.

Results

We trained our model on COCO-2017 object detection dataset (117k images) on Google Cloud (NVIDIA A100 40GB VRAM, 12-Core CPU, 85GB RAM) for 4 epoch with batch size 48 (resulting in total of approximately 10k iterations). We managed to achive comparable results to FasterRCNN (with ResNet50 backbone) which was trained for ~ 300k iterations. The loss and metrics (mAP and AP across scales) are improving over the epochs so we believe that more training would lead to better results.

Model	mAP	AP_{small}	AP_{medium}	AP_{large}
FasterRCNN	34.9	15.6	38.7	50.9
GATor	32.2	13.3	31.6	43.6

We used SGD + Momentum optimizer with learning rate of 0.02 and weight decay of 0.0005 and 1000 warmup iterations. We also experimented with Adam and RAdam optimizers which proved to be suboptimal for the object detection task. All images in training dataset are resized to 640x480 before processing.

Improvements

One improvement would be to add global contextual information in form of virtual nodes. The algorithm would have two steps – using Louvain algorithm to compute graph partitions, then adding virtual node and connecting it to all the nodes in its partition. The partition nodes would then be connected to a single global node.

Second improvement would be to create a new dataset from COCO2017 which contains only graphs created from images. This would allow for faster GAT training since no CNN backbone is needed, but the downside is the dataset size (~ 1TB).

Another possible improvement would be to use different edge weights and different edge types (e.g. semantic edges and spatial edges).