Proposal

Automatically Generated

Lenny Brown Caroline Rodewig

1 Introduction

We describe and propose a novel algorithm for automatic SRNN generation and optimization that is fully differentiable end-to-end and really cool. This assumes some previous knowledge of the SRNN paper Jain et al. (2015).

2 Method

2.1 Image Segmentation / Object Detection

We first implement and train (or use pre-trained) an extremely fast object recognition (Ren et al. (2015)) or image segmentation (Shelhamer et al. (2016)) neural network. Our output for this is such that for every pixel, we would like $\{X_{i,j} \mid \sum_{x \in X_{i,j}} x = 1\}$ where $X_{i,j}$ is a vector representing a distribution for some pixel at index i, j over possible segments or objects in our picture that that pixel could belong too. The resulting tensor is $X \in \mathbb{R}_{N \times M \times K}$ where N is the width, M is the height, and K is the number of objects we want. This is extremely useful because it allows us to treat the structural part of SRNNs as a fluid (and learned). If something enters the frame, we can just recognize it and add it to one of the object layers. This might push some other things into a different layer also, but the most relevant fact is that this can be done in an optimized, pre-set form which allows us to make use of existing frameworks.

2.2 NodeRNN

This rank-3 tensor X is the input for our RNN. The only difference between this and when we use a matrix is that instead of one matrix to describe (for instance) the forget gate, we have k matrices, each of which describes the forget gate weights for a single object $\{o \in X_k \mid X_k \in \mathbb{R}_{N \times M}\}$. Boiled down to its essence, this is no different then the NodeRNNs of Jain et al. (2015) except this setup can be automatically generated and trained simultaneously.

2.3 EdgeRNNs

This setup also conveniently lends itself to the automatic creation and optimization of EdgeRNNs, though, since they are very different from Jain et al. (2015), it is *slightly* disingenuous to name them such. Our method treats the edge problem as a convolutional one, wherein we represent each edge between two objects (in the st-graph) as a convolution layer(s) with dimensions $i, j, 2$. The dimensions indicate the fact that we will be stacking

the object layers on top of each other and the convolution layer will look for interaction(s if multiple) between the two objects. The benefits of this setup are that interactions between objects can be learned which are invariant of position (a useful property, or so I'm told) and that it maintains continuous differentiability. The total number of possible tuples is incalculable (not formally, I'm just too lazy right now its something like $2^{n(n-1)/2}$).

2.3.1 Guess

Though not the direction of this paper because we have to prove this method works, future work could look at automatically generated (and learned) setups wherein some administrating RNN decides the proper number and placement of "edges".

2.4 Output

The above methodology is extremely flexible and thus, it would be silly to describe how all of it would fit together in an absolute sense. If we concatenated the output all the RNNs and performed a generalized dot product with some weight matrix, then we would end up with a fully connected network. If we keep them seperate, we could attempt to translate each individual EdgeRNN, which would be much easier because we know the two (or more) objects involved and thus can talk only about their interactions. Then, we could rank all of the outputs of this kind of method by some importance metrix in order to describe the most relevant aspects of the scene. However, since we are indeed *proposing* a task for ourselves, we propose two different tests. First, we would recreate (in a basic sense, just for feasibility) the tests done in Jain et al. (2015), which of course implies that our end connections will be "wired" like theirs (i.e. the edges added together for input into the nodes). Second, if we show that we our methodology is indeed a feasible recreation, we would demonstrate an extension of it which attempts to continuously label videos.

References

Jain, A., A. R. Zamir, S. Savarese, and A. Saxena, 2015: Structural-rnn: Deep learning on spatio-temporal graphs. *CoRR*, **abs/1511.05298**.

Ren, S., K. He, R. B. Girshick, and J. Sun, 2015: Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, **abs/1506.01497**.

Shelhamer, E., J. Long, and T. Darrell, 2016: Fully convolutional networks for semantic segmentation. *CoRR*, **abs/1605.06211**.