Notes from Human Shape Code

Arjun Gupta

September 6, 2019

0.1 General Notes on Graph Networks

Many of these notes are from reading this paper (https://arxiv.org/pdf/1812.08434.pdf). Graph NN's can be considered a generalization of CNN's to non-euclidean space. Graph NN's were made to exploit the local substructure of graphs much like CNN's were made to exploit the local substructure of images.

The paper argues the CNNs and RNNs cannot accurately be used for graphs because they consider an ordering of nodes (pixels in the CNN case), however graphs do not have a particular order and should be order invariant. There are several forms of GNNs, including:

1. Graph Convolutional Networks
2. Graph Attention Networks
3. Graph Spatial-Temporal Networks
4. Graph Auto-Encoders
5. Graph Generative Networks

0.2 Notes on Human Shape Code

The core of their model is defined as `graph_cnn` which has three main parts:

1. `gc` - The majority of the graph convolution network.
2. `shape` - The final section of the graph convolution network for getting the SMPL shape.
3. `camera_fc` - The small model to estimate the camera pose.

`gc` is the majority of the model as far as the graph convolutions go. The input to `gc` are the reference vertices of the SMPL mesh concatenated with the output of passing the image through ResNet-50. `gc` is the "trunk" network which processes the vertices and the image, and then `shape` returns the non-parametric down-sampled SMPL Mesh.

`gc` consists of what they define as a `graph_linear` block followed by several of what they call `graph_residual` blocks. The `graph_linear` block is effectively just a matrix multiplication of the feature vectors for each of the graph nodes with a weight matrix and is meant to correspond to a 1×1 convolution in a CNN. The `graph_residual` block is far more complicated. It contains the following in order:

1. Group Normalization
2. Graph Linear
3. Group Normalization

4. Graph Convolution
5. Group Normalization
6. Graph Linear

It then adds the result of passing the input through those layers back to the input to get the final result. The Group Normalizations are all followed by ReLU non-linearities. The authors decide to use Group Normalization rather than the standard Batch normalization because they observed Batch Normalization to achieve poor performance and No-normalization was slow and prone to local minima. The core of the graph residual block is the Graph Convolution layer which is based on the paper here (https://arxiv.org/pdf/1609.02907.pdf). The convolution described by the paper has the following form:

$$f(X, A) = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}XW\right) \tag{1}$$

Where A is the adjacency matrix, $\tilde{A} = A + I_N$ (i.e. the graph with self-connections added), $\tilde{D}_{ii} = \sum_j A_{i,j}$, and W is the weights matrix. However, based on the code and what is written in the paper, it appears that the convolution really has the following form:

$$f(X, A) = AXW \tag{2}$$

Upon further inspection, it seems that $A = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ which is the Normalized Graph Laplacian.

The network outputs the 3d coordinates of each node in the output mesh.