Medal RnD

Learning spatial relationships in images using graph convolutions

Shalabh Gupta, Abhinav Kumar, Parth Dodhia

{180050095, 180050003, 180070041}

Advisors : Prof Amit Sethi, Prof Suyash Awate

1 Introduction/Problem Description

Convolutional neural networks are a class of deep learning algorithms known for analysing visual imagery. These networks efficiently capture high and low level image features and are thus widely used in image related tasks like classification, segmentation, object detection to name a few. Their recent resurgence has been primarily due to the performance of AlexNet [1] which famously won the ImageNet challenge in 2012.

The power of CNN comes from the fact that they have much less parameters and connections when compared to similar sized neural networks because of sharing among the image patches, along with neighbourhood capturing receptive fields across many layers. However, even with this efficiency, convolutional networks have been expensive to apply in large scale high resolution images [1].

Another inability of convolution networks comes from them being unable to incorporate global context beyond local regions [2]. To capture long range relationships, we may have to apply many layers of convolutions.But with increasing number of layers, resolution of images decrease along with loss of low level details and increase in number of parameters. Therefore convolution and pooling operations are not much suitable for coding intricate spatial relationships between distant objects in an image. [2] argue that even after many layers of convolutions, the effective receptive field of a network's units is severely limited.

Through this project, we try to approach this problem of coding long distant relationships by finding equivalent graph representations of images which contain information sufficient to reconstruct back the images. Graph convolutions are further applied upon the graph representations to finally obtain feature vectors that can be used to perform tasks such as classification and segmentation. We train and validate our models on primarily two datasets, the tinyimagenet dataset and a subset of ImageNet dataset (containing 10 classes).

2 Why Graphs?

Graphs are natural choice for coding spatial relationships because of their message passing tendency amongst the vertices. Imagine the vertices of this graph as cluster of regions in the original image. With this propagation of information between regions, we are able to incorporate global context in individual region feature vectors and we believe that this property may help reconstruct the images from the graphs as well.

Our method basically consists of learning relevant interest points in an image which will further act as vertices for the graph representation. We build the edge features (which is basically the adjacency matrix) by two different methods, as we will show in the corresponding section. Then we apply graph convolutions on these graph structures, which update vertex features based on the adjacency matrix and finally a fully connected network applied on the feature vector helps us classify. The model is end-to-end trainable.

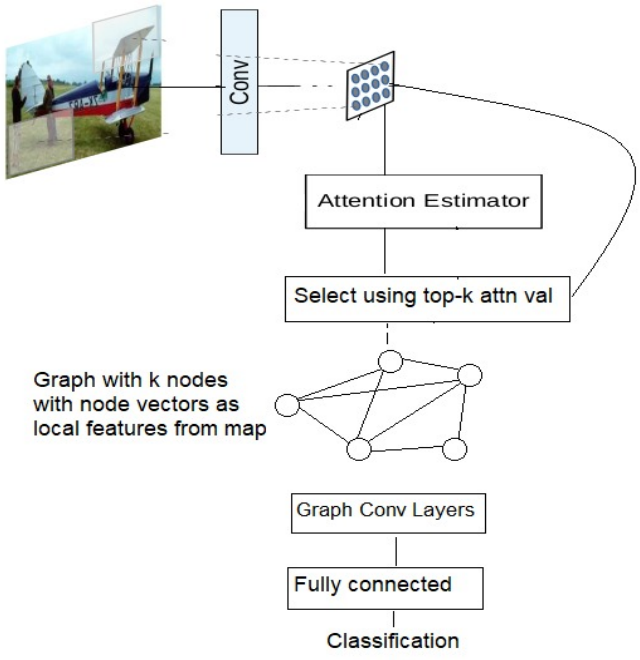

Figure 1: Main Model Flow

3 Capturing Interest points

A graph is composed of vertices and edges. The vertices can be interest points, that are detected during inference that follows a learning process. Before Deep Learning became popular, interest points were defined as points on an image plane that were locally unique, such as at a corner.

3.1 Unsupervised clustering

To achieve interest point detection in an unsupervised setting, [2] employ a soft clustering based method, where pixels are mapped to vertices in the graph using a matrix Q, where $Q[i, j, k]$ denotes the assignment measure of pixel (i, j) to k. Naturally, as in other soft clustering mechanisms, $\sum_k Q[i, j, k] = 1$. Weights W are learnt during training, where w_k i.e. $W[k]$ act as anchor points for each of the vertices and help assigning the pixels to vertices according to similarity scores. Finally vertex features are taken to be simply a weighted average of feature maps of all pixels assigned to the given vertex.

3.2 Attention Score mechanism

One approach to visualising and interpreting the inner workings of CNNs we came up with is the attention map: a scalar matrix representing the relative importance of layer activation at different 2D spatial locations with respect to the target task. This notion of a nonuniform spatial distribution of relevant features being used to form a task-specific representation, and the explicit scalar representation of their relative relevance, is what we term 'attention'. Previous works have shown that for a classification CNN trained using image-level annotations alone, extracting the attention map provides a straightforward way of determining the location of the object of interest
And thus we believe that using attention scores can provide us with interest points, as high attention score at a position can generally and intuitively relate to an object of high relevance when compared to the global context of the image. Now we describe mechanisms employed by us to find attention scores in our models.

3.2.1 Model1

This attention mechanism is inspired from the approach used in Learn to Pay Attention [3]. This approach, originally, was based on the hypothesis that there is benefit to identifying salient image regions and amplifying their influence and likewise suppressing influence on non-useful regions. Due to a dot product with the global context vector, this method learns the relative importance of positions with respect to the whole image which intuitively act as attention scores.
The method consists of finding compatibility scores between feature maps in intermediate layers and the global context vector obtained upon the application of fully connected layers post convolution. Formally, the compatibility function C takes input l_i^s, the feature vector of pixel i from layer s, and $\mathbf{g}$ (the global vector), returing a compatibility

score c_i^s as can be seen in the image below. These scores serve as attention score for ith pixel, and the top k scoring pixels are chosen as interest points. This is done for 3 intermediate layers, and thus a total of $3 * k$ points are chosen as nodes in the graph. The compatibility score is calculated as shown below.

$$c_i^s = \langle \mathbf{l_i^s}, \mathbf{g} \rangle \tag{1}$$

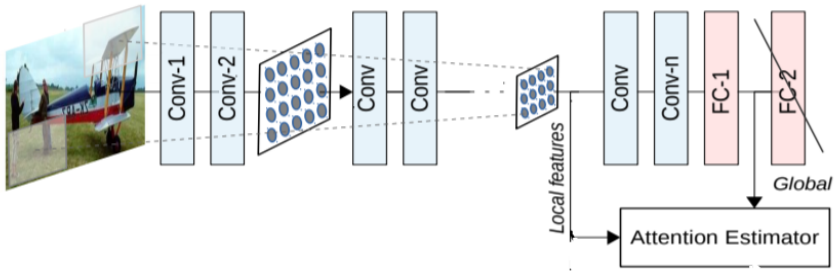

Figure 2: Attention Mechanism from [3]

3.2.2 Model2

The above described approach uses the global vector **g** for calculating the attention scores but it relies on the output of deep convolutional layers, but we want to reduce this dependency and let the graph convolutions do the major heavy lifting as aimed earlier due to shortcomings of convolutions. So, here we propose another approach inspired from the DELF (Deep local features) block based attention mechanism from [4] to obtain attention scores through a single convolution operation on the feature maps obtained from the backbone. This is as straightforward as it gets and easily end to end as can be seen in Fig.1 where, after applying the convolution, we obtain scalar attentions.

3.2.3 Model3

This idea is the extension of the previous approach in which we use a residual block instead of a convolution layer with the belief that a more complex attention mechanism would be able to capture the intuition better. The attention block is trainable along side the main architecture in unsupervised setting as previously described.

4 Graph Convolutional Networks description

Graph Convolution make use of the graph structure, and convolve based on method suggested by [5]. This helps in updating vertex features according to the adjacency matrix already formed.

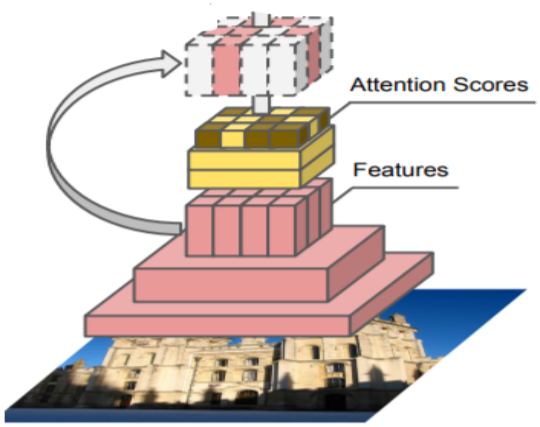

Figure 3: Attention Mechanism from [4]

4.1 Matrix formation

The first step before applying graph convolution is formation of the adjacency matrix, which inherently stores the edge features, since A[i,j] denotes a scalar value that is the some measure of similarity (or distance, for that matter) between nodes i and j. We discuss two methods of matrix formation.

4.1.1 Similarity Measure

Owing to how [2] and [6] build their adjacency matrix, we can simply stack the vertex features as columns into a matrix, say X, and calculate the adjacency matrix to be as follows:

$$A = X^T X \tag{2}$$

Here any element $A[i, j] = \langle X[i], X[j] \rangle$, which can be thought of as a similarity measure between feature vectors of node i and j, thus a node i to j relation.

4.1.2 Euclidean Distance

This method depends on encoding adjacency matrix in form of inverse euclidean distances. The intuition behind this approach is that the nodes in this graph setting are the pixels (or superpixels), and edges represent spatial distances between them [7]. That is, $A[i, j]$ will be a factor of inverse distance between node i and node j where the nodes are those pixels who had the highest attention scores. Note that inverting the distance is important since closer nodes should result in stronger relations i.e. higher $A[i, j]$ value.

4.1.3 Normalization

Before being subjected to a weight matrix, we have to aggregate node features from all neighboring nodes. This is typically done by multiplying layer activations by the

adjacency matrix, as we will see in the next subsection. But directly multiplying by the adjacency obtained above has a few limitations [8]. First is that the aggregated representation does not contain self features. This is fixed by adding the identity matrix to A. The second major limitation is that A is typically not normalized and therefore multiplying by A will completely change the scale of the feature vectors, leading to issues like exploding gradient and other problems while training. For this, [5] suggested a symmetric normalization method, $D^{\frac{-1}{2}}AD^{\frac{-1}{2}}$ where D is the diagonal node degree matrix. Formally:

$$A = A + I \tag{3}$$

$$\hat{A} = D^{-0.5}AD^{-0.5} \tag{4}$$

4.2 Convolution

We follow the following layer wise propagation rule:

$$X^{(l+1)} = \sigma(\hat{A}X^{(l)}W^{(l)}) \tag{5}$$

Here $\hat{A}$ is the normalized adjacency matrix as described above in subsection 4.1.3, $X^{(l)}$ are the activations in layer l being the input to layer $l+1$, $W^{(l)}$ is the trainable weight matrix for layer l and finally $X^{(l+1)}$ are the activations in layer $l+1$. $\sigma(.)$ just denotes a non linear activation, and we have used it to be ReLU. Multiplication of $X^{(l)}$ with $\hat{A}$ here incorporates the sum of all feature vectors over all neighboring nodes for each node as discussed earlier. Since applying it once pools over first order neighbourhood, applying it repeatedly helps in propagation of information throughout the graph, which is our aim.

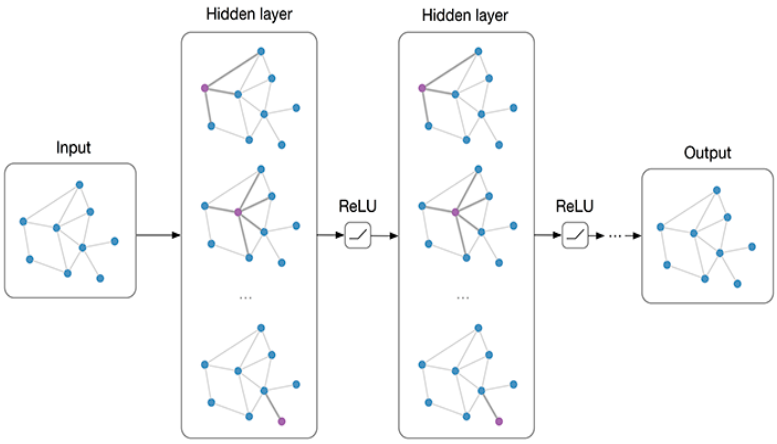

Figure 4: GCN from [5]

5 Fully Connected Network

After the node feature vectors have been processed by Graph Convolution layers, their sample-wise mean is passed to a multi layer perceptron for classification task and the hypothesis is that as node vectors are the representation of the interest points in the image, their mean should be able to capture most of the information necessary for prediction.

5.1 MLP details

We feed the mean representation obtained earlier to a Multi-Layer fully connected network at the end of which is a softmax layer with required number of classes for prediction. We tried different configurations for the layers and different number of layers too to observe its effect on prediction and whether a deeper network can help for the same input representation.

6 Model details

We tried different convolutional backbones obtaining the initial feature maps as well as different architectures for obtaining the attention scores for this feature map. Also there are two different ways we form our adjacency matrix as discussed above. We now present details of three major variations we tried.

6.0.1 Model1

The first model is inspired from VGG network except some pooling layers have been moved so that the feature maps (shown in the figure below) are of size sufficient to select k vectors having highest attention scores for graph formation.
Here the attention score mechanism is as described in section 3.2.1 with the initial conv layers resembling VGG architecture. Deeper layers will be able to capture global interest points better and shallower layers will be more dependent on local features. So, we try a mix of deep and shallow feature vectors to build the graph. For tiny-imagenet dataset (64×64 images), we select $k = 40$ nodes with highest attention scores from each of successive feature maps of size $32 \times 32, 16 \times 16, 8 \times 8$, all having feature vectors of length 256. These $3 \times k = 120$ nodes form the vertices of the graph and their feature vectors are fed to 3 consecutive GCN layers (Section 4). Adjacency matrix here is formed using the similarity measure (subsection 4.1.1). Then the mean node representation is fed to the MLP layer for prediction over 200 classes.

6.0.2 Model2

This model has a pretrained resnet50 architecture instead of the vgg-like architecture in the previous model. A pretrained model usually has better configuration and information with it than a non pretrained one, and thus converges faster with better

classification results. Rather than learning this backbone convolution model alongside the main GCN part training, we use a pretrained resnet50 based architecture in both trainable and non-trainable settings to achieve better results. Also, rather than extracting feature maps from multiple layers, we use the feature map from a single layer only. The reason for this is discussed later. The attention mechanism used in this model is the same as Model2 in section 3.2.2 which uses *DELF* block. For tiny-imagenet dataset, we select $k = 40$ feature vectors (size 512) having highest attention score from 8×8 feature map from intermediate resnet50 layer. The adjacency matrix here again is using the similarity measure (subsection 4.1.1) and 3 layers of GCN with output features of size 256. Then mean representation fed to MLP for classification as described above.

6.0.3 Model3

The backbone model for obtaining feature maps is the same resnet50 architecture as above but we have changed the attention model from 1 layer convolution to more complex residual block like network (Section 3.2.3). We trained this model on multiple datasets and on comparison with previous models, it fared better than both of them. For training, we used tiny-imagenet(200 classes 64×64 images) and 2 other subsets of imagenet (10 classes with 224×224 images). The size of the feature map, from which top-k features are selected, are $h/8 \times w/8$ where $h \times w$ is the size of input image and the value of k is 40 and 120 respectively for tiny-imagenet and the other subset of imagenet. Here we use the euclidean distance method to form the adjacency matrix (subsection 4.1.2). The GCN layers used in this model are same as the above 3-layered network.

7 Evaluation

Dataset	Evaluation	Model1	Model2	Model3	Resnet
Tiny-Imagenet	Train	0	68	54	96
	Test	0	40	44	55
Imagenett-1	Train	NA	96	95	98
	Test	NA	83	88	92
Imagenett-2	Train	NA	81	92	96
	Test	NA	62	75	88

Table 1: Performance of resnet50 along with various models we experimented with

We found that on tiny-imagenet dataset with many classes, all our models overfit at some point of time which we believe is due to the small size of the image and partly because of many classes to be classified among and too less data in that context. The same problem of overfitting does not occur for the two imagenet-subsets (224×224) with 10 classes thus strengthening our hypothesis. Our model with euclidean distance matrix and complex attention mechanism (Model 3) performs the best among the three models as expected.

Figure 5: Dice Loss(loss) vs epochs for all three axes (4 class segmentation)

8 Challenges/Problems faced

- Our initial models faced a major problem of overfitting on tiny imagenet dataset, despite having very less parameters. As described above, it was mainly because dataset is too less to account for 200 classes in it.
- Also, we think that while the idea of attention scores to find interest points is promising theoretically, it does not work practically because of the limitation of the global context being involved and difficulty of gradient flow through locations.
- The first attention method we tried went deeper into the convolution layers to find the global vector, which we would branch back to help formulate attention scores. But this violated our aim of not providing much space to convolutions.
- However, with the second attention method, global context is not taken into account and it is left to the network to learn the attention scores according to the relevance of points to the task. This is problematic because letting the gradients flow back from locations is imperative to this, but we could not find a way to do so. Instead, for a lot of time, we were stuck on the debugging of gradient flow problem in our network, which we finally resolved in a temporary fashion by multiplying attention scores with the feature vectors obtained from the backbone.
- Another issue with model 1 that we tried is that duplicate positions (pixels) may be captured as interest points since we select from three different layers.
- Also, usually attention scores in a neighbourhood of a high scored pixel are high. This is as expected due to continuacy in images. This will lead to interest points being very close to each other, but we require sparse interest points to capture more image related information.
- Overall, we faced a lot of challenges, and learnt a lot throughout the course of the project.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. Curran Associates Inc., 2012.

[2] Yin Li and Abhinav Gupta. Beyond grids: Learning graph representations for visual recognition. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 9225–9235. Curran Associates, Inc., 2018.

[3] Namhoon Lee Philip H. S. Torr Saumya Jetley, Nicholas A. Lord. Learn to pay attention. In *International Conference on Learning Representations*, 2018.

[4] Hyeonwoo Noh, Andre Araujo, Jack Sim, and Bohyung Han. Image retrieval with deep local features and attention-based keypoints. *CoRR*, abs/1612.06321, 2016.

[5] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[6] Mohammed Adnan, Shivam Kalra, and Hamid R Tizhoosh. Representation learning of histopathology images using graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 988–989, 2020.

[7] Boris. Tutorial on graph neural networks for computer vision and beyond, Aug 2017.

[8] Tobias Skovgaard. How to do deep learning on graphs with graph convolutional networks, Jan 2019.