Efficient Relative Attribute Learning using Graph Neural Networks

Zihang Meng[1], Nagesh Adluru[1], Hyunwoo J. Kim[1⋆],
Glenn Fung[2], and Vikas Singh[1]

[1] University of Wisconsin – Madison
[2] American Family Insurance
zihangm@cs.wisc.edu, adluru@wisc.edu,
hwkim@cs.wisc.edu, gfung@amfam.com, vsingh@biostat.wisc.edu

Abstract. A sizable body of work on relative attributes provides evidence that relating pairs of images along a continuum of strength pertaining to a visual attribute yields improvements in a variety of vision tasks. In this paper, we show how emerging ideas in graph neural networks can yield a solution to various problems that broadly fall under relative attribute learning. Our main idea is the observation that relative attribute learning naturally benefits from exploiting the graph of dependencies among the different relative attributes of images, especially when only partial ordering is provided at training time. We use message passing to perform end to end learning of the image representations, their relationships as well as the interplay between different attributes. Our experiments show that this simple framework is effective in achieving competitive accuracy with specialized methods for both relative attribute learning and binary attribute prediction, while relaxing the requirements on the training data and/or the number of parameters, or both.

Keywords: Relative attribute learning, graph neural networks, multi-task learning, message passing

1 Introduction

Visual attributes [6] correspond to mid-level semantic and even non-semantic concepts or properties of the image or objects contained in the image that are interpretable by humans. For instance, an image can be "natural", "smiling" or "furry" depending on the properties of the key entities contained in it. The ability to associate such attributes with images has enabled systems to perform better in traditional categorization tasks, and even go beyond basic level naming [18]. The insight in this line of work is to first select features that can predict attributes for the object class of interest – the subsequent classifier must then leverage only those "relevant" features since material properties or shape may be differentially important for different categories. The concept of "relative attributes" takes this idea further [18] by arguing that the strength of an attribute in an image is best

⋆ work performed during studies at UW-Madison in 2017, before joining Amazon judged in the context of its strength with respect to all other images in the training data rather than as a binary concept. For example, while it is difficult to characterize how "man-made" an image is, one could setup a comparison where humans compare the images in terms of *this* attribute. This strategy of describing images in relative terms works well in challenging cases [13] – for instance, calculating how "open" an image is versus another.

Since the early works on relative attributes [21, 20, 23], several papers have proposed more task-specific models for ranking based on specialized features. But given the success of convolutional neural networks (CNN) architectures, most recent proposals utilize CNNs for feature learning in the context of learning the overall ranking. For instance, given a set of annotated image-pairs with respect to one/more attributes, the network learns weights that are maximally consistent with the attribute-specific ranking of the images. Related ideas have also explored designing image-part specific detectors, that are aligned to an attribute. For instance, what is the spatial support for an attribute such as "smiling". Clearly, this will involve localizing the visual concept to a part of the image, say the mouth or lips region. In [20], the authors transitively connect the visual chains across the attribute continuum and make the case that feature extraction and ranking should not be performed separately.

The starting point of our work is the observation that the space of attributes which induce a ranking over the images share a great deal of correlational structure. For instance, the attribute "furry" may be associated with the attribute "four-legged" and the attribute "congested" may have some information to provide to the attribute "man-made". This induces a natural graph of attributes and images, where the input data provides either pair-wise relationships between images for one/more attributes or a partial (or full) ranking of the images for the attribute. We do not assume that the annotation is exhaustive – many edges (or relationships) between the images may, in fact, be unavailable. Extending recent work on graph neural networks (GNNs) which extends the notion of convolution and other basic deep learning operations to non-Euclidean grids ([11, 19, 16, 9, 3]), we show how these ideas yield a natural model for learning on this graph involving image$\leftrightarrow$attribute and image$\leftrightarrow$image edges. Not only are the image features (relevant for each attribute) extracted automatically but we also concurrently learn the similarity function that is most consistent with the given pair-wise annotations as well as the latent relationships between the attributes (similar to multi-task learning). This machinery is simple, yet performs competitively with more specialized proposals on several different problems.

Our **contributions** are: (1) we formulate and solve relative attribute learning via a message passing scheme on a graph, where the convolutional layers, ranking as well as imputation of unseen relationships is performed concurrently. (2) our framework yields results similar to the best reported for each task with minimal change, often providing sizable reduction in the number of parameters to be estimated or with far less stringent requirements on the training data annotations. We note that GNNs were independently used in a classification task very recently in a paper made available on arXiv [8].

2 Related Work

Visual attributes. Visual attributes are semantic properties in images which can be understood by humans and are shared among all images of similar categories (e.g. all images of human faces share the attribute "smiling", whose strength can vary from weak to strong as we will show with examples shortly. Most existing work in visual attributes focuses on binary attribute prediction (BAP) where each attribute is predicted from a given image and cast as a binary classification problem. "Relative attributes" were proposed in [18] and have been explored in a number of settings [20, 21, 23]. Several current techniques use deep neural networks to learn relative attributes (e.g. [21]), and also borrow ideas from attention mechanism research (e.g. [20]) to help the networks focus only on the most informative areas in the images. Most of these works deal with a pair of images at a time. Our work shows that dealing with groups of images on a fully connected graph instead of just pairwise comparisons improves performance.

Multi-task learning. Multi-task learning is intended to achieve knowledge sharing by learning several correlated tasks at the same time. This technique has recently been used in binary attribute prediction. Learning several correlated attributes together can improve performance, and this has been demonstrated by some recent works [1, 12, 22]. Abdulnabi et al. [1] propose a multi-task CNN framework which improves accuracy compared with learning one attribute at a time. Wang et al. [22] designed a simpler deep multi-task network for prediction of face attributes. In contrast to most strategies related to multi-task learning, our multi-task formulation learns attributes simultaneously and is shown to benefit relative attribute learning.

Graph neural networks (GNN). Graph neural networks were proposed by [11, 19], where the authors describe GNN as a parameterized message passing scheme which can be trained. Later, Li et al. [16] proposed using gated recurrent units (GRUs) within GNNs, which much improves the representation capacity of the network and makes it suitable for graph structured data. Gilmer et al. [9] generalized the GNN using message passing neural network and demonstrated state-of-the-art results on molecular prediction benchmarks. More recently, concurrent to and independent of our work, [8] applied GNNs for classification and achieved good results on several different datasets.

3 Approach

Our approach is based on the observation that in a relative attribute learning task, different images are correlated and the attributes may or may not be correlated. The learning procedure can benefit from exploring the similarity among multiple images on a graph, where each node represents an image and the edges are formed based on the relationship between the to-be-learned representations of the nodes. Furthermore, such a graphical structure can benefit multi-task learning where we can add different *types* of nodes to the graph for representing different attributes that are being learned. In this way, we explicitly learn the properties of certain attributes, the interplay between the attributes when necessary, the representations of the images and their relationships on the graph in a way that best informs the task at hand.

We first explain how the input images are mapped into the graph representation, and give the details of our network architecture for relative attribute learning in the context of *one attribute*. Then, we show how the construction can be used to perform multi-task attribute learning with minimal modifications. Finally, we also show how our model can be used for a binary attribute prediction (BAP) task efficiently. The overview of our framework is shown in Fig. 1.

3.1 Network Architecture

Let $\mathcal{I} = \{I_1, I_2, \cdots, I_n\}$ be the set of input images, and for a certain attribute t (e.g., smile), we assume that a set of pairwise relationship labels $\mathcal{P}_l^t = \{\phi(I_i, I_j)\}_{i,j=1;i\neq j}^n$, where $\phi(I_i, I_j)$ indicates the relative strength of the attribute t between the two images I_i and I_j. This relationship may be logical (e.g.,"stronger than" or "weaker than"). With this data, a generalized GNN is trained where both the node features (representations of the images) and edge weights are learned. The core architecture of our GNN is shown in Fig. 2.

Assume that we operate on groups (or mini-batches) of a certain size (which is allowed to vary) sampled with or without replacement from the underlying training dataset. The relationships among all the images in each mini-batch (S) in the training set are represented using a fully-connected graph $G_S = (V, E)$, where each node v_i in V corresponds to an image I_i in the mini-batch

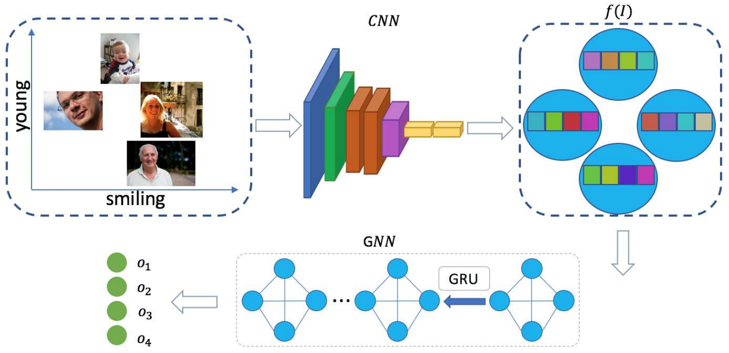

Fig. 1. Overview of our framework for RAL and BAP tasks. Since many natural attributes of images are interrelated, discovering their common latent representations would be beneficial to the attribute learning tasks. This can be efficiently achieved by mapping these images to a graphical structure. Every image has a corresponding node on the graph and a corresponding output node. The initial features $f(\cdot)$ for the nodes are generated using a CNN on the images and the edge features and following updates are performed using GNNs (details in Fig. 2). The weights in the entire framework including those in the CNN and GNN are trained end-to-end.

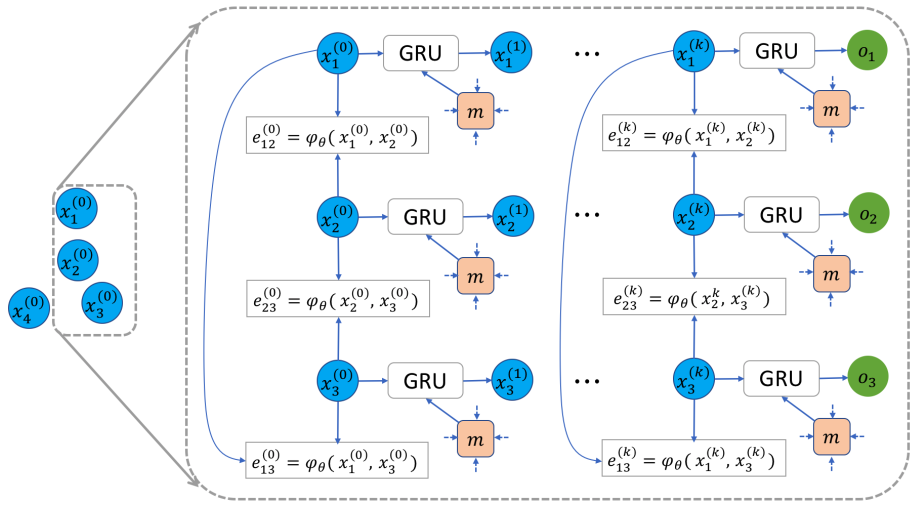
GRU
m
$e_{12}^{(0)} = \varphi_\theta(x_1^{(0)}, x_2^{(0)})$
$e_{23}^{(0)} = \varphi_\theta(x_2^{(0)}, x_3^{(0)})$
$e_{13}^{(0)} = \varphi_\theta(x_1^{(0)}, x_3^{(0)})$
$e_{12}^{(k)} = \varphi_\theta(x_1^{(k)}, x_2^{(k)})$
$e_{23}^{(k)} = \varphi_\theta(x_2^{k}, x_3^{(k)})$
$e_{13}^{(k)} = \varphi_\theta(x_1^{(k)}, x_3^{(k)})$
o_1
o_2
o_3

Fig. 2. The architectural details of our GNN which remains the same for both RAL and BAP. The edges on the graph are learned from adjacent nodes using a parameterized function (φ_ϑ, see (3)), which is shared among all edges. The "m" in this figure refers to the message for a node passed from its connected nodes and edges, which is defined in (4). Then, a GRU cell takes as input a node and its corresponding message, and outputs the updated node. The parameters in GRU are also shared across all nodes.

S. Each time, the network takes in a group of images and passes them through a convolutional neural network. This may also be thought of as a set of $|S|$ convolutional networks that share weights. The representations derived from this network yield the initial representations of the node features as

$$x_i^{(0)} = f(I_i), \tag{1}$$

where $f(\cdot)$ refers to a CNN which operates on the images. Here, I_i is the input image and $x_i^{(0)}$ is the initial node feature for the image at time $k = 0$. Next, the network learns edge features as,

$$e_{i,j}^{(k)} = \varphi_\vartheta\left(x_i^{(k)}, x_j^{(k)}\right), \tag{2}$$

where φ is a symmetric function parameterized with a single layer Neural Network:

$$\varphi_\vartheta(x_i^{(k)}, x_j^{(k)}) = \mathfrak{N}_\vartheta^{\text{edges}}\left(||x_i^{(k)} - x_j^{(k)}||_1\right). \tag{3}$$

We assume that φ_ϑ is a metric which is learned by a non-linear combination of the absolute difference between the learned features of the two nodes (or any other simple function involving the node features). This ensures that the symmetric property $\varphi_\vartheta(a, b) = \varphi_\vartheta(b, a)$ is satisfied by design.

Our goal now is to update the belief at each node based on the beliefs at the other nodes in the graph as well as its own state at the previous time point.

To accomplish this, we use a message function $M(\cdot)$ to aggregate information from all neighbors of each node. In particular, for each node $x_i^{(k)}$, the message is defined as below,

$$m_i^{(k)} = \sum_{j, j \neq i} M\left(x_j^{(k)}, e_{i,j}^{(k)}\right). \tag{4}$$

Here, $M(\cdot)$ is parameterized using a single layer neural network whose details are presented later in section 3.2. We now need to define a mechanism that utilizes the messages received from the node's neighbors and its previous state to update its state. To do so, we use an updating layer $\mathrm{G}(\cdot)$ which takes as input a signal $x^{(k)}$ (the current state of the node) and produces $x^{(k+1)}$. This is accomplished using a Gated Recurrent Unit (GRU) as the updating function.

$$x_i^{(k+1)} = G\left(x_i^{(k)}, m_i^{(k)}\right). \tag{5}$$

With this setup in hand, we simply use a readout function $o_i = R(x_i)$ to get the output from each node and finally define our loss function based on these outputs from all relevant nodes on the graph as

$$\text{Loss} = \mathcal{R}\left(\{o_i\}_{i=1}^{n}\right), \tag{6}$$

where n is the number of graph nodes. Note that $\mathcal{R}(\cdot)$ can also be parameterized with a simple (or more complicated) neural network depending on the needs of the application. The specific form of $\mathcal{R}(\cdot)$ depends on the concrete task, which will be specified in the following Sections 3.2–3.3.

3.2 Learning Relative Attributes, One at a Time

The Relative Attribute Learning (RAL) task seeks to learn a network that, given input images, outputs pairwise labels according to the relative strength of certain attributes between each pair of images. In this section, we consider training a network for one attribute at a time.

Recall that our network is designed to better explore the correlated information among different images. So unlike other approaches in RAL ([21, 20]) which take two images at a time as an input, we sample a group of images from the training set as input at every draw. The size of the group need not be fixed and can vary for learning different attributes in a single dataset or different datasets, because our network has the benefit of weight sharing on the graphical structure of the samples. We use the five convolutional layers and the first two fully-connected layers in `AlexNet` [15] (`conv1` through `fc7`) although other architectures can be substituted in. The dimension of the output feature vector of the node is fixed to be 4096.

Messages. We impose a fully-connected graphical structure on the images in each group. After mapping these images on the graph, we perform message passing, which is effective in information propagation among the nodes. We adopt the strategy to *learn* edge features from the current node hidden representation formulated by Gilmer et al. [9], as suggested in (2). The parameters of the edge learning function φ_ϑ are shared among all nodes on the graph. Then for every node $x_i^{(k)}$ on the graph, a message signal will be extracted from all the in-coming nodes through the edges, see (4). Here, we specify the message function $M(\cdot)$ as,

$$M(x_j^{(k)}, e_{i,j}^{(k)}) = \mathrm{ReLU}\left(W\left(x_j^{(k)} \| e_{i,j}^{(k)}\right) + b\right), \tag{7}$$

where $\|$ denotes the concatenation operator of two vectors, W and b are the weight matrix and the bias respectively, and $\mathrm{ReLU}(\cdot)$ is the rectified linear unit (ReLU) function. We would also like to note that the parameters (W and b) of the message function $M(\cdot)$ are also shared by all nodes and edges in our graph, thus providing an explicit control on the number of parameters.

Updating. Let us now discuss the updating function for nodes. At each iteration, each GRU takes the previous state of the node and an incoming message as input, and produces a new hidden state as the output (see Fig. 2). Let $x_i^{(k-1)}$ be the node's hidden representation at the previous time step, $m_i^{(k)}$ be the message received via (4), and $x_i^{(k)}$ be the updated node. With these notations, the basic operations of GRU are simply given as,

$$\begin{aligned} z_i^k &= \sigma\left(W^z m_i^{(k)} + U^z x_i^{(k-1)}\right), \\ r_i^k &= \sigma\left(W^r m_i^{(k)} + U^r x_i^{(k-1)}\right), \\ \tilde{x}_i^{(k)} &= \tanh\left(W m_i^{(k)} + U\left(r_i^k \odot x_i^{(k-1)}\right)\right), \\ x_i^{(k)} &= (1 - z_i^k) \odot x_i^{(k-1)} + z_i^k \odot \tilde{x}_i^{(k)}, \end{aligned} \tag{8}$$

where z and r are the intermediate variables in the GRU cells, $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid function and $\odot$ is element-wise multiplication.

Each node in our graph maintains its internal state in the corresponding GRU, and all nodes share the same weights of the GRU, which makes our model efficient while can also seamlessly deal with differently sized groups as input. In this work, we use one time step of GRU updating. During testing time, any number of images are allowed, and the network will output a pairwise label for every two images based on the obtained value of output nodes on the graph. After constructing our graph using (1)–(6), the loss defined on the output of graph takes the form

$$\mathrm{RALLoss} = \sum_{i,j,i\neq j} -\mathcal{L}\log(P_{ij}) - (1-\mathcal{L})\log(1-P_{ij}), \text{where} \tag{9}$$

$$\mathcal{L} = \begin{cases} 1 & \text{if } I_i \succ I_j, \\ 0 & \text{if } I_i \prec I_j, \\ 0.5 & \text{otherwise}, \end{cases}$$

and $P_{i,j} = o_i - o_j$ (outputs of nodes i and j). This formulation has a nice property that it is robust to noise as described in [4], and symmetric by construction so that we can easily utilize training data where some pairs of images appear with "equal" label for one/more attributes.

3.3 Learning Relative Attributes, All at Once

In this section, we show that our graphical structure can be efficiently applied to learn multiple relative attributes all at the same time i.e., perform multi-task attribute learning. We consider two aspects of multi-task learning, (1) the performance of RAL can be improved by utilizing several attributes which have common latent representations. Although this has been demonstrated in binary attribute prediction (BAP) setting, we present experimental results showing that RAL can benefit from multi-task learning. (2) The second aspect is the efficiency of the construction. While multi-task learning can improve the performance when attributes are correlated, in the previous methods [1, 22], the number of parameters of the network grows *much faster* as a function of the number of attributes learned together, which increases the cost of training a multi-task network. As an example, if the number of parameters trained in RAL one at a time is $O(K^2)$ then our version only increases the number to $O(K^2+nK)$, where n is the number of different relative attributes learned simultaneously. This is much smaller than $O(nK^2)$ which may be needed within other multi-task approaches [1, 22].

We note that a näive way to adapt our network (Fig. 2) to the multi-task setting proceeds as follows. We simply change the dimension of the output o_i from 1 to m where m is the number of attributes. But the only change this induces is in the size of the weight matrix in the readout function. We find that in this case, the graphical structure may slightly lose its expressive capacity. To address this issue, unlike section 3.2, which treats all nodes in the graph in the same way, here, we define two different *types* of nodes x_i, $i = 1, 2, \cdots, n$, and r_i, $i = n+1, n+2, \cdots, n+m$, where n is the number of input images in each group (to be consistent, we choose $n = 5$ throughout our experiments) and m equals the number of attributes the network is learning at the same time. Here, x_i has the same meaning as in section 3.2, which corresponds to one image and each r_j corresponds to a certain attribute. It is important to note that while the representation at x_i is learned by the convolutional network, the attribute node r_j is randomly initialized at the beginning of the training phase and keeps getting updated in a global manner, similar to the other parameters in the GNN.

This scheme allows us to explicitly learn a hidden representation for each attribute in a way that the latent variables of the graphical model are influencing all attribute nodes — this is similar to multi-task learning where we expect that learning related tasks can benefit each other when carried out concurrently. The feature extraction process using the convolutional network and the GNN procedure remain identical as in section 4.1. The only change needed is to redefine how we use the readout function $R(\cdot)$ to get the output. Here, $o_{i,j} = R(||x_i - r_j||_1)$, where for $o_{i,j}$, i gives the index of nodes for the images (from 1 to n) and j gives the index of different attributes (from 1 to m). The loss function is then defined as the sum of the loss for each single attribute (see (9)) as,

$$\mathrm{RALLoss}_{\mathrm{multi}} = \sum_{i=1}^{m} \mathrm{RALLoss}_i. \tag{10}$$

3.4 Binary Attribute Prediction

In this section, we present details of how our graphical model can also be used to predict binary attributes with comparable accuracy as the multi-task CNN model [1], but using much fewer number of parameters.

Binary attribute prediction (BAP) task seeks to predict whether an image has a certain attribute (e.g. whether a person is wearing a necktie), which can be thought of as a binary classification task. As suggested in papers for multi-task learning [1, 22], simultaneously learning several attributes which are correlated can improve the performance of BAP. In this setting, the labels no longer provide pairwise information. So, it is not simple to easily extend other RAL methods and adapt them for BAP. For example, the construction using Siamese network [20] cannot be easily modified for BAP since the subnetworks are no longer linked – this is because it is the pairwise annotations that link the networks. But our network can still benefit from a fully-connected graph structure on the training samples because despite the unavailability of pairwise annotations, the images themselves are still related. So, we can use the *same* basic architecture. The framework before loss layer remains the same as the network in section 3.3. The loss function for BAP is simply defined as

$$\mathrm{BAPLoss}_i = -\mathcal{L}\log(P_i) - (1-\mathcal{L})\log(1-P_i), \tag{11}$$

where $\mathcal{L}$ is the binary label of image I_i, and $P_i = o_i$. The total loss is defined as,

$$\mathrm{BAPLoss}_{\mathrm{multi}} = \sum_{i=1}^{m} \mathrm{BAPLoss}_i. \tag{12}$$

4 Experimental Results

In this section, we analyze the performance of our model on several different settings described in section 3. First, we present some key **implementation details.** Our network takes in a group of images and outputs the pairwise relationships for this group (in a relative attribute task) or a binary label for each image (in a attribute prediction task). We split the train/test set randomly. Then, we randomly split the train/test set into groups (we choose 5 images per group, but the number can vary) and use this as the input to our network. We report the pairwise accuracy measured on the groups of images. In a preprocessing step, we subtract the mean of training set and crop images to size 227×227.

For training, we initialize the `conv1` to `fc7` layers using `AlexNet` pre-trained on the ILSVRC 2012 [15] dataset and randomly initialize other parts using the Xavier initializer [10]. We use mini-batches of size 10 and Adam optimizer [14] with $\beta_1 = 0.9$, $\beta_2 = 0.999$. The learning rate of relative attribute learning task is 0.0001, and for attribute prediction task, we set the learning rate to 0.00001.

4.1 Relative Attribute Learning, One at a Time

In this experiment, we evaluate the network described in section 3.2. The goal is to compare pairs or sets of images according to the strength of a given attribute. We used the OSR scene dataset [17] and a subset of the Public Figure Face Dataset (PubFig) [18]. The OSR scene dataset consists of 2,688 images with outdoor scene attributes (`natural`, `open`, `perspective`, `large-objects`, `diagonal-plane` and `close-depth`). The subset of the PubFig contains nearly 800 images from 8 random identities. We split the train/test set randomly and then split the train/test set into groups and use this as the input of network. We report the results in terms of pairwise accuracy on the groups of images.

Our scheme makes it possible to make use of the information in the group of images as a whole, which is more informative than just a pair of images (common to Siamese Networks construction). For a fair comparison with other methods, we measure the performance of our model by computing the pairwise accuracy for all pairs in each group.

We choose two methods for baseline comparisons. The first one is the work of Souri et al. [21], which trains a deep convolutional network to learn relative attributes for pairs of images. The second one is the DeepPermNet [2], which learns relative attributes by learning permutations. Note that this method needs fully ranked sequences of images as input, which is a more stringent requirement compared to our network and the work of Souri et al. [21], which only needs pairwise labels during training. The accuracy results are shown in Tables 1–2. Qualitative results are shown in Fig.3.

Table 1. Relative attribute learning accuracy evaluated on **OSR** dataset. On average, we outperform all previous methods. The penultimate row presents the results of our network in section 4.1 and the last row presents the results of our multi-task network in section 4.2, which learns all of the six attributes at once.

Method	`natural`	`open`	`perspective`	`large-objects`	`diagonal-plane`	`close-depth`	**Mean**
Souri et al.[21]	99.4	97.44	96.88	96.79	98.43	97.65	97.77
Cruz et al.(AlexNet)[2]	97.21	96.65	96.46	98.77	94.53	96.09	96.62
Cruz et al.(VGG)[2]	96.87	99.79	99.82	99.55	97.99	96.87	98.48
Ours	99.56	99.19	99.30	98.08	**99.63**	97.98	98.96
Ours(multi-task)	**99.89**	99.42	98.71	98.80	99.46	**98.93**	**99.20**

Compared to the work of Souri et al. [21], we outperform that method by a margin of 4% on the Public Figure Face Dataset, and by 1% on the OSR scene dataset. Since the accuracy on the OSR dataset is already high, a 1% improvement is meaningful. Compared with the DeepPermNet [2] algorithm, we outperform that algorithm on both datasets on average. Note that DeepPermNet requires *ranked sequences* of data with the same length as training data, which may not be possible in some applications. Also note that both Souri et al. [21] and DeepPermNet [2] use `VGG CNN` model in their experiments, while we choose the simpler `Alexnet` [15] in all experiments, which has far fewer parameters. As a result, our model can be trained faster than the baseline models.

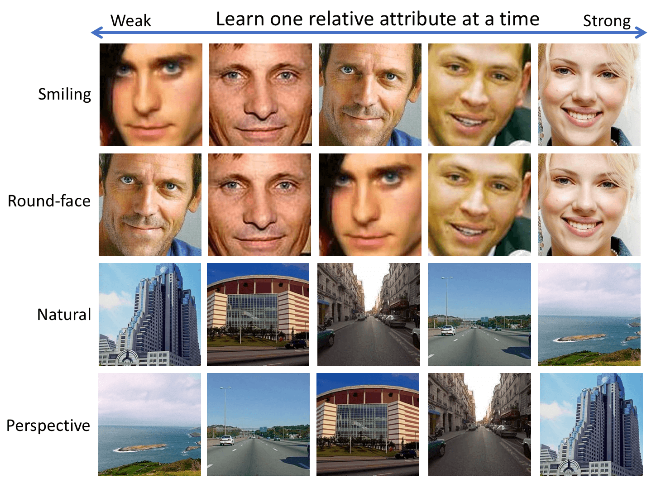

Fig. 3. Qualitative results on RAL (**one at a time**) from our network. We randomly choose five different images from four different attributes from the PubFig and OSR datasets and show the results by ordering them for those attributes. The images are ranked by the corresponding output value of our network. The first two rows are from the PubFig dataset and the last two rows are images from the OSR dataset.

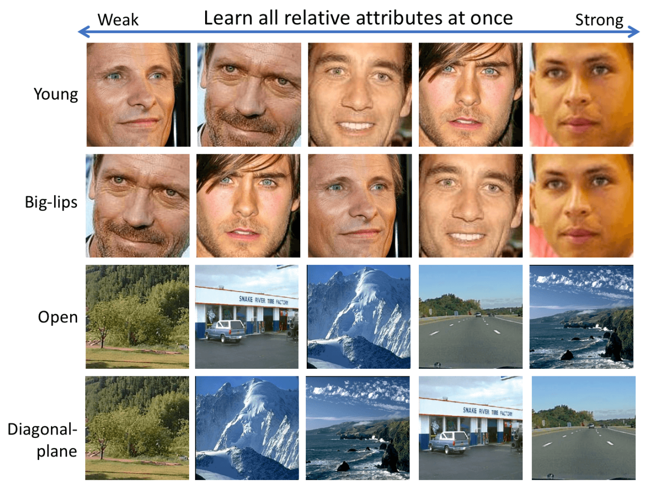

Fig. 4. Qualitative results on RAL (**all at once**) using our network. The images are arranged again by ordering them according to the output of our network as Fig. 3 but these are learnt from our multi-task loss function (Eq. (10)). We can see that the images are quite nicely ordered even without learning the order explicitly as is done in DeepPermNet. We also note that the performance on almost all the images and the attributes is consistent and any randomly chosen subset gives us good quality results.

Table 2. Relative attribute learning accuracy evaluated on the **PubFig** dataset. Our results outperform the work of Souri et al. [21], which is the state-of-art for the traditional setting where only pairwise labels are used. Our results are also competitive and get slightly better results than those in Cruz et al. [2], which uses ranked input data. The last row shows the results of our network with multi-task loss function, which learns all of the 11 attributes at once.

Method	lips	eyebrows	chubby	male	eyes	nose	face	smiling	forehead	white	young	**Mean**
Souri et al.[21]	93.62	94.53	92.32	95.59	93.19	94.24	94.76	95.36	97.28	94.60	94.33	94.52
Cruz et al.[2]	99.55	97.21	97.66	99.44	96.54	96.21	99.11	97.88	99.00	97.99	99.00	98.14
Ours	98.28	97.11	98.67	98.05	**98.62**	**99.24**	97.32	**99.26**	98.37	**99.36**	**99.31**	98.51
Ours(multi-task)	**99.67**	**99.33**	**99.00**	98.33	97.32	98.46	99.00	97.51	**99.12**	97.66	98.66	**98.55**

4.2 Relative Attribute Learning, All at Once

In this experiment, we evaluate our multi-task network described in section 3.3. We learn all the attributes in each dataset and report the prediction accuracy results for each of the attributes on two different datasets in Table 1 and 2. Qualitative results are shown in Fig. 4.

As the data presented show, our multi-task model slightly outperforms our single attribute learning model (section 3.2) and this indicates that some of the attributes are interrelated thus helping the learning process when we learn them all at once. Note that in our framework, with every additional attribute to learn, the increase in the number of parameters of the network is equal to the dimension of the two vectors, one in the readout function and one in the attribute node (in our work, the dimension of these two vectors is 4096×1). The reader may contrast this with most multi-task learning networks, such as [22, 12, 1], many of which use an additional CNN or several more fully connected layers for each additional attribute, which contribute to more parameters compared to our model.

4.3 Binary Attribute Prediction

Here we evaluate our network for attribute prediction task described in section 3.4. The multi-task CNN model [1] is a natural choice for the baseline. This model proposes to pre-train a convolutional neural network on each attribute to get the feature vectors, and then performs multi-task learning for multiple attributes. That model has a large number of parameters and a rich representation capacity. Similar to [1], we also evaluate our model on Clothing Attributes Dataset [5]. It contains 1,856 images and 26 attributes. The ground truth is provided at the image-level, and each image is annotated for every attribute. For comparison, we ignore the multi-class value attributes as in [1] and use this information in the same way to divide the 23 binary attributes into groups. We then use our multi-task network to train each group of attributes together. We report our results in Table 3 and the group information is provided in Table 4. M-CNN is the multi-task framework without group information in [1] and MG-CNN is their multi-task framework with group encoding. The performance of our

Table 3. Attribute prediction accuracy on the Clothing Dataset [5]. Similar to [1] we partition the 23 binary attributes into 4 groups (shown in Table 4). We achieve comparable results as those from MG-CNN [1], but with significantly fewer parameters (see section 3.3) and faster training speed.

Method	**Colors**	**Patterns**	**Cloth-parts**	**Appearance**	**Total**
M-CNN[1]	91.72	94.26	87.96	91.51	91.70
MG-CNN[1]	93.12	95.37	88.65	91.93	92.82
Ours	91.64	96.81	89.25	89.53	92.39

model is comparable to the results presented in MG-CNN framework, but is far more efficient both in the number of parameters and convergence time. For the number of parameters, [1] needs one CNN for each attribute, while we only add 4096×1 parameters twice. In terms of training time, MG-CNN[1] takes 1.5 days for training on the Clothing dataset with two NVIDIA TK40 16GB GPU, while our training takes less than 4 hours for all 4 groups of attributes on two NVIDIA Geforce GTX 1080Ti 12GB GPU.

Table 4. Grouping information used in Clothing Dataset[5]

Group	**Attributes**
Colors	`black`, `blue`, `brown`, `cyan`, `gray`, `green`, `many`, `red`, `purple`, `white`,`yellow`
Patterns	`floral`, `graphics`, `plaid`, `solid`, `stripe`, `spot`
Cloth-parts	`necktie`, `scarf`, `placket`, `collar`
Appearance	`skin-exposure`, `gender`

4.4 Limitations

For our network to get a sizable performance benefit, we want that the graph formed by each random sample, i.e., group or mini-batch of n (e.g., $n = 5$) images should be "connected" or at least a subgraph with more than 2 nodes is connected. This allows learning from more than one image pair at a time to be meaningful – which is the main strength of our proposal. But if most pair labels do not have any node overlap, then the graph formed by a group or mini-batch of images will not have a connected component of size larger than two. We refer the reader to [7] (Chapter 4) to see the technical aspects of connectivity. The UT-Zappos50K dataset [24, 25] manifests this behavior (and is not ideal for our model to deliver performance gains). Under this condition, our model actually performs similar to (although not exactly the same) a Siamese network used in the literature. The results in Table 5 indeed support this intuition: our performance is only slightly better than [21], rather than stronger improvements we see elsewhere.

Table 5. Relative attribute learning evaluated on UT-Zappos50K-lexicon dataset. It contains 50025 images of shoes with annotations on 4000 ordered pairs for each of 10 fine-grained attributes. The method in [2] does not directly work on this dataset because of its "ordered sequence" requirement on the input data.

Method	comfort	casual	simple	sporty	colorful	durable	supportive	bold	sleek	open	**Mean**
Souri et al.[3]	88.93	89.20	88.27	91.33	91.67	89.27	91.00	88.40	88.27	86.80	89.31
Ours	88.80	**89.82**	**90.13**	**92.60**	**91.87**	**90.07**	**92.73**	88.00	87.53	**89.13**	**90.07**

5 Conclusions

We presented a simple framework that can perform both relative attribute learning and attribute prediction. To exploit the underlying relationships between latent representations of a variety of attributes among a collection of images in a dataset, we proposed a simple framework based on natural instantiation of graph neural network. This formulation of a graph neural network can effectively encode the correlational information among multiple images and the multiple attributes as demonstrated in our experiments on three different datasets. Our framework can be used to learn the relative attributes either one at a time or all at once with only a modest increase in the number of parameters compared to other multi-task based methods. Because our framework learns mainly from pairs of images and does not require a full ranking it concurrently is less stringent on the annotation requirements of the training dataset. To the best of our knowledge, this proposal is among the first to explore the efficacy of multi-task GNN formulations for relative attribute learning. Our experiments also demonstrate the effectiveness of this architecture in achieving or surpassing the state-of-the-art results even for binary attributes prediction, where each attribute is predicted in a binary classification setup. The project webpage includes results on other applications, including predicting body mass index (BMI) that were not covered in detail in the main paper.

6 Acknowledgment

This work was partially supported by funding from American Family Insurance and UW CPCP AI117924. Partial support from NSF CAREER award 1252725, NIH grants R01 AG040396, BRAIN Initiative R01-EB022883, and Waisman IDDRC U54-HD090256 is also acknowledged. The authors are grateful to Haoliang Sun for help with illustrations and other suggestions/advice on this project. The code will appear in `https://github.com/zihangm/RAL_GNN`.

References

1. Abdulnabi, A.H., Wang, G., Lu, J., Jia, K.: Multi-task cnn model for attribute prediction. IEEE Transactions on Multimedia **17**(11), 1949–1959 (2015)
2. Anoop, R.S.C.B.F., Gould, C.S.: Deeppermnet: Visual permutation learning. learning **33**, 25

3. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine **34**(4), 18–42 (2017)
4. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: Proceedings of the 22nd international conference on Machine learning. pp. 89–96. ACM (2005)
5. Chen, H., Gallagher, A., Girod, B.: Describing clothing by semantic attributes. In: European conference on computer vision. pp. 609–623. Springer (2012)
6. Farhadi, A., Endres, I., Hoiem, D., Forsyth, D.: Describing objects by their attributes. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. pp. 1778–1785. IEEE (2009)
7. Frieze, A., Karoński, M.: Introduction to random graphs. Cambridge University Press (2015)
8. Garcia, V., Bruna, J.: Few-shot learning with graph neural networks. arXiv preprint arXiv:1711.04043 (2017)
9. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. arXiv preprint arXiv:1704.01212 (2017)
10. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. pp. 249–256 (2010)
11. Gori, M., Monfardini, G., Scarselli, F.: A new model for learning in graph domains. In: Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on. vol. 2, pp. 729–734. IEEE (2005)
12. Han, H., Jain, A.K., Shan, S., Chen, X.: Heterogeneous face attribute estimation: A deep multi-task learning approach. IEEE transactions on pattern analysis and machine intelligence (2017)
13. Jamieson, K.G., Jain, L., Fernandez, C., Glattard, N.J., Nowak, R.: Next: A system for real-world development, evaluation, and application of active learning. In: Advances in Neural Information Processing Systems. pp. 2656–2664 (2015)
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
16. Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.: Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493 (2015)
17. Oliva, A., Torralba, A.: Modeling the shape of the scene: A holistic representation of the spatial envelope. International journal of computer vision **42**(3), 145–175 (2001)
18. Parikh, D., Grauman, K.: Relative attributes. In: Computer Vision (ICCV), 2011 IEEE International Conference on. pp. 503–510. IEEE (2011)
19. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Transactions on Neural Networks **20**(1), 61–80 (2009)
20. Singh, K.K., Lee, Y.J.: End-to-end localization and ranking for relative attributes. In: European Conference on Computer Vision. pp. 753–769. Springer (2016)
21. Souri, Y., Noury, E., Adeli, E.: Deep relative attributes. In: Asian Conference on Computer Vision. pp. 118–133. Springer (2016)
22. Wang, F., Han, H., Shan, S., Chen, X.: Deep multi-task learning for joint prediction of heterogeneous face attributes. In: Automatic Face & Gesture Recognition (FG 2017), 2017 12th IEEE International Conference on. pp. 173–179. IEEE (2017)

23. Xiao, F., Jae Lee, Y.: Discovering the spatial extent of relative attributes. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1458–1466 (2015)
24. Yu, A., Grauman, K.: Fine-grained visual comparisons with local learning. In: Computer Vision and Pattern Recognition (CVPR) (Jun 2014)
25. Yu, A., Grauman, K.: Semantic jitter: Dense supervision for visual comparisons via synthetic images. In: International Conference on Computer Vision (ICCV) (Oct 2017)