Deep Multi-Task Augmented Feature Learning via Hierarchical Graph Neural Network

Pengxin Guo[1], Chang Deng[3], Linjie Xu[4], Xiaonan Huang[1], and Yu Zhang (✉)[1,2]

[1] Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China
`12032913@mail.sustech.edu.cn,xnhuang92@outlook.com,yu.zhang.ust@gmail.com`
[2] Peng Cheng Laboratory, Shenzhen, China
[3] Committee on Computational and Applied Mathematics, University of Chicago, Chicago, USA
`changdeng@uchicago.edu`
[4] Game AI Group, Queen Mary University of London, London, UK
`linjie.xu@qmul.ac.uk`

Abstract. Deep multi-task learning attracts much attention in recent years as it achieves good performance in many applications. Feature learning is important to deep multi-task learning for sharing common information among tasks. In this paper, we propose a Hierarchical Graph Neural Network (HGNN) to learn augmented features for deep multi-task learning. The HGNN consists of two-level graph neural networks. In the low level, an intra-task graph neural network is responsible of learning a powerful representation for each data point in a task by aggregating its neighbors. Based on the learned representation, a task embedding can be generated for each task in a similar way to max pooling. In the second level, an inter-task graph neural network updates task embeddings of all the tasks based on the attention mechanism to model task relations. Then the task embedding of one task is used to augment the feature representation of data points in this task. Moreover, for classification tasks, an inter-class graph neural network is introduced to conduct similar operations on a finer granularity, i.e., the class level, to generate class embeddings for each class in all the tasks using class embeddings to augment the feature representation. The proposed feature augmentation strategy can be used in many deep multi-task learning models. Experiments on real-world datasets show the significant performance improvement when using this strategy.

Keywords: Multi-task Learning · Feature Learning · Graph Neural Network.

1 Introduction

Multi-task learning [8,38] aims to leverage useful information contained in multiple learning tasks to improve their performance simultaneously. During past decades,

Corresponding author: Yu Zhang (`yu.zhang.ust@gmail.com`).

many multi-task learning models have been proposed to identify the shared information which can take a form of the instance, feature, and model, leading to three categories including instance-based multi-task learning [5], feature-based multi-task learning [3, 19, 21, 23, 26, 28, 31, 41], and model-based multi-task learning [2, 6, 11–14, 17, 39, 40].

At present, using the output of shared hidden layer as the representation of hidden features shared by tasks is the mainstream approach in deep multi-task learning, which has achieved good results in many problems. However, these methods can not learn the task-specific feature representation of each task, which limits the further improvement of the performance. At another extreme, some works [9, 26] use a neural network as a base model for each task in multi-task learning. One advantage of this approach is that different tasks can learn their own feature representation, however, a major problem is that the model parameters of the entire multi-task learning model are linear with respect to the number of tasks, making this approach not scalable to a large number of tasks in multi-task learning.

In this paper, we study deep multi-task learning between the two extremes and hope to learn task-specific feature representation to improve the learning performance but without increasing the number of parameters as well as the model complexity too much. To achieve that, we propose to use the task representation, which is also called the task embedding, as a type of an augmented feature representation to improve the expressiveness of the feature representation as well as the performance, since the task embedding contains the unique characteristics of a task. To derive the task embedding of each task, the training dataset of that task is used in terms of a graph where nodes represent data points and edges denote the similarities between data points. Besides, the relationship between tasks can also be represented in a graph. Inspired by this idea, in this paper, we propose a Hierarchical Graph Neural Network (HGNN) to further improve the performance of multi-task learning models by learning augmented features. The HGNN consists of two-level graph neural networks. In the first level, an intra-task graph neural network is to learn a powerful representation for each data point in a task by aggregating its neighbored data points in this task. Based on the representation learned in the first level, we can generate the task embedding, which is a representation for this task, in a way similar to max pooling. For classification tasks, we can generate the class embedding for each class in this task based on max pooling. Based on task embeddings of all the tasks generated in the first level, an inter-task graph neural network in the second level updates all the task embeddings based on the attention mechanism. For classification tasks, an inter-class graph neural network is introduced in the second level to update all the class embeddings based on neighbored class embeddings. Finally, each of the learned task embeddings as well as the class embeddings for classification tasks is used to augment the feature representation of all the data points in the corresponding task. The proposed HGNN can be used in many multi-task learning models. We analyze the use of HGNN in terms of both the training loss and generalization loss. Extensive experiments show the effectiveness of the proposed HGNN. Our contributions are summarized as follows:

– We propose a Hierarchical Graph Neural Network (HGNN) to learn augmented features for deep multi-task learning. The proposed feature augmentation strategy can be used in many deep multi-task learning models for regression and classification tasks.
– We provide some analyses for the proposed feature augmentation strategy, which can help understand why the incorporation of the task embedding can improve the performance.
– We conduct extensive experiments on four benckmark datasets to demonstrate that HGNN can improve the performance of many deep multi-task learning models.

2 Related works

Liu [20] explores the problem of learning the relationship between multiple tasks dynamically and formulate this problem as a message passing process over a graph neural network. Meng [25] solves relative attribute learning via a message passing scheme on a graph and the main idea is that relative attribute learning naturally benefits from exploiting the dependency graph among different relative attributes of images. The multi-task attention network proposed in [21] consists of a single shared network containing a global feature pool and a soft-attention module for each task that allows to learn task-specific feature-level attentions. Lu [24] presents a graph star net which utilizes the message-passing and attention mechanisms for multiple prediction tasks, including node classification, graph classification, and link prediction. Though the aforementioned works propose GNN or the attention mechanism for multi-task learning, none of them use a hierarchical version of GNN as well as the attention mechanism to learn augmented features for multi-task learning, which is the focus of this paper.

Kim [15] proposes a hierarchical attention network for stock prediction which can selectively aggregate information on different relation types and add the information to each representation of the company for the stock market prediction. However, after obtaining the additional information to each representation, this work only uses the neighbored nodes to aggregate the information, which is different from our work that aggregates all the nodes in the graph. Moreover, this method adds the additional information to the original feature representation, which is different from the concatenation method used in this paper. Ryu [30] proposes a Hierarchical graph Attention-based Multi-Agent actor-critic (HAMA) method, which employs a hierarchical graph neural network to effectively model the inter-agent relationships in each group of agents and inter-group relationships among groups, and additionally employ inter-agent and inter-group attentions to adaptively extract state-dependent relationships among agents. However, similar to [15], the HAMA method, a network stacking multiple Graph Attention Networks (GAT) [33] hierarchically, only processes local observations of each agent but not all the information in the graph. Different from these two methods, we first use an intra-task graph neural network to generate a task embedding for each task by using all the data in this task and then use the inter-task graph neural network to update task embeddings of all the tasks based on the inter-task structure. To the best of our knowledge, we are the first to use the feature augmentation strategy in multi-task learning.

3 Hierarchical Graph Neural Network

In this section, we introduce the proposed architecture, the Hierarchical Graph Neural Network (HGNN), for deep multi-task learning. Whilst the architecture can be incorporated into any multi-task learning network, in the following sections we show how to build the HGNN upon a multi-task network.

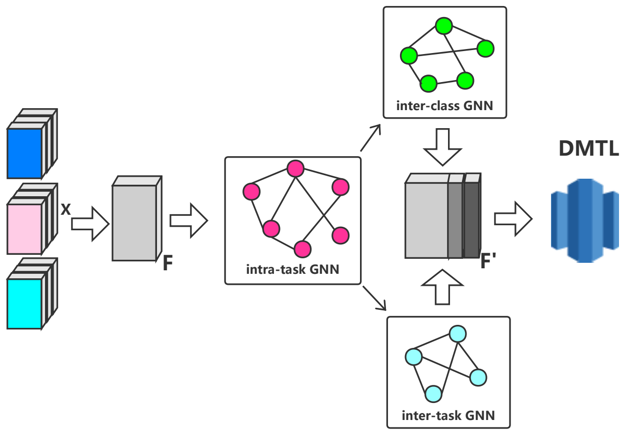


Fig. 1. An illustration of the hierarchical graph neural network for multi-task learning, where **F** is the hidden feature representation, 'intra-task GNN' is the first level GNN to aggregate all the information contained in the data of a task to generate the task embedding, 'inter-task GNN' and 'inter-class GNN' (for classification task) are the second level GNN to update all the task embeddings and class embeddings by sharing information among all the tasks and classes, and $\mathbf{F}'$ is the augmented feature representation used to do prediction. $\mathbf{F}'$ is the concatenated of the hidden feature representation **F** and its corresponding task embedding and class embedding (for classification task).

3.1 Overview of The Architecture

The HGNN consists of two-level GNNs. The first-level GNN is an intra-task GNN to aggregate all the information contained in the data of a task to generate a task representation, which is called the task embedding. In the second level, based on the generated task embeddings in the first level, an inter-task GNN is used to update all the task embeddings by sharing information among all the tasks. Finally the task embeddings are used to augment the feature representation of the data to improve the learning performance. For classification tasks, we can learn augmented features in a fine granularity - the class level. That is, the intra-task GNN is also used to aggregate all the information in a class of a task to generate a class embedding. Then based on class embeddings in all the tasks, an inter-class GNN is used to update them. Finally, both task embeddings and class embeddings are used to augment the feature representation. The whole architecture of HGNN is shown in Figure 1.

3.2 The Model

Suppose that there are m multi-class classification tasks where each task has k classes. The training dataset of the ith task consists of n_i pairs of data samples and corresponding labels, i.e., $\mathcal{D}^i = \{(\mathbf{x}_p^i, y_p^i)\}_{p=1}^{n_i}$ where $y_p^i \in \{1, \dots, k\}$.

For $\mathbf{x}_p^i$, we first define its hidden representation as

$$\hat{\mathbf{h}}_p^i = \sigma_s(\hat{\mathbf{W}}_s \mathbf{x}_p^i + \hat{\mathbf{b}}_s), \tag{1}$$

where $\sigma_s(\cdot)$ can be any activation function such as the ReLU function, and $\hat{\mathbf{W}}_s, \hat{\mathbf{b}}_s$ are shared parameter among all the tasks. Eq. (1) defines the shared layer in a multi-task neural network.

For the intra-task GNN, we first construct an adjacency matrix $\mathbf{G}^i$ for the ith task based on the hidden representation and label information. Specifically, the (p, q)th entry in $\mathbf{G}^i$, g_{pq}^i, can be defined as

$$g_{pq}^i = \begin{cases} \exp\{-\|\hat{\mathbf{h}}_p^i - \hat{\mathbf{h}}_q^i\|_2^2\} & \text{if } y_p^i = y_q^i \\ -\exp\{-\|\hat{\mathbf{h}}_p^i - \hat{\mathbf{h}}_q^i\|_2^2\} & \text{otherwise} \end{cases},$$

where $\|\cdot\|_2$ denotes the ℓ_2 norm of a vector. Then the intra-task GNN can be defined as

$$\mathbf{H}^i = \sigma_h(\mathbf{W}_h^i \mathbf{X}^i + \hat{\mathbf{H}}^i \mathbf{G}^i + \mathbf{b}_h^i \mathbf{1}), \tag{2}$$

where $\sigma_h(\cdot)$ can be any activation function such as the ReLU function, $\mathbf{X}^i = (\mathbf{x}_1^i, \dots, \mathbf{x}_{n_i}^i)$, $\hat{\mathbf{H}}^i = (\hat{\mathbf{h}}_1^i, \dots, \hat{\mathbf{h}}_{n_i}^i)$, $\mathbf{1}$ denotes a vector of all ones with an appropriate size, $\mathbf{W}_h$ and $\mathbf{b}_h$ are the parameters in the GNN. $\mathbf{G}^i$ in Eq. (2) can make similar data points in the same class have similar representations in $\mathbf{H}^i$ and dissimilar data points from different classes have dissimilar representations. The intra-task GNN can have two or more layers each of which is defined as in Eq. (2).

Based on the intra-task GNN, the task embedding of the ith task is defined as

$$\mathbf{e}_t^i = \max_p \{\mathbf{h}_p^i\},$$

where the max operation is conducted elementwisely, $\mathbf{h}_p^i$ is the pth column in $\mathbf{H}^i$. So the task embedding is obtained via the max pooling on all the data points in the ith task based on the hidden representation learned by the intra-task GNN. Similarly, the class embedding of the rth class in ith task is defined as

$$\mathbf{e}_c^{i,r} = \max_{p:y_p^i=r} \{\mathbf{h}_p^i\},$$

which means that the class embedding of the rth class in the ith task is obtained via the max pooling on all the data points in the rth class of the ith task based on the intra-task GNN. We have tried other pooling methods such as the mean pooling but the performance is inferior to the max pooling. One reason is that the max pooling can bring some nonlinearity but the mean pooling is a linear operation.

Then m task embeddings $\{\mathbf{e}_t^i\}_{i=1}^m$ for the m tasks can form a graph. The inter-task GNN is responsible of learning for the graph constructed by task embeddings $\{\mathbf{e}_t^i\}_{i=1}^m$ to generate new task embeddings $\{\hat{\mathbf{e}}_t^i\}_{i=1}^m$ by exchanging information among tasks. Here we use GAT as an implementation of the inter-task GNN. In order to learn powerful task embeddings based on the inter-task relation, each task embedding $\mathbf{e}_t^i$ is first transformed by a weight matrix $\mathbf{W}$. Then we perform *self-attention* on the task embeddings. That is, an attentional mechanism computes *attention coefficients* as

$$d_{ij} = a(\mathbf{W}\mathbf{e}_t^i, \mathbf{W}\mathbf{e}_t^j),$$

where the attentional mechanism $a(\cdot,\cdot)$ we use is the cosine function, which is different from the original GAT. To normalize coefficients, we transform d_{ij} via the softmax function as

$$\alpha_{ij} = \text{softmax}_j(d_{ij}) = \frac{\exp(d_{ij})}{\sum_l \exp(d_{il})}.$$

Attention values can be viewed as a measure of task relations between each pair of tasks. Once obtained, the normalized attention coefficients are used as combination coefficients to compute the updated task embeddings via a nonlinear activation function σ as

$$\hat{\mathbf{e}}_t^i = \sigma\Big(\sum_{j=1}^m \alpha_{ij}\mathbf{W}\mathbf{e}_t^j\Big).$$

According to this equation, we can see that $\hat{\mathbf{e}}_t^i$ contains useful information from embeddings of other tasks. In experiments, the inter-task GNN adopts two such layers to generate the new task embeddings.

Similarly, the mk class embeddings $\{\mathbf{e}_c^{i,r}\}$ also can form a graph. We use another inter-class GNN to generate new class embeddings $\{\hat{\mathbf{e}}_c^{i,r}\}$ in a similar way to the inter-task GNN.

The learned task embeddings and class embeddings can be used to augment the data feature representation to form a more expressive one as $\tilde{\mathbf{h}}_p^i = \text{concat}(\hat{\mathbf{h}}_p^i, \hat{\mathbf{e}}_t^i, \hat{\mathbf{e}}_c^{i,r})$, where concat$(\cdot,\cdot,\cdot)$ denotes the concatenation operation. Then data in such augmented representation can be fed into a deep multi-task learning model to predict class labels. To see that, the objective to be minimized in our proposed learning framework for multi-task learning can be formulated as

$$\mathcal{L} = \sum_{i=1}^{m} \frac{1}{n_i} \sum_{p=1}^{n_i} \mathcal{L}_C(f(\mathrm{concat}(\gamma(\mathbf{x}_p^i), \hat{\mathbf{e}}_t^i, \hat{\mathbf{e}}_c^{i,y_p^i})), y_p^i) + \lambda r(\boldsymbol{\Theta}), \quad (3)$$

where $\mathcal{L}_C$ denotes the classification loss on the available labeled data, $f(\cdot)$ denotes the learning function of a multi-task neural network starting from the second layer, the function $\gamma(\cdot)$ denotes the function defined in Eq. (1) by omitting the parameters, $r(\boldsymbol{\Theta})$ denotes a regularization function on $\boldsymbol{\Theta}$ that includes all the parameters of the model, and λ is a regularization parameter.

3.3 Testing Process

At the testing process, we do not know the true label, hence we cannot directly concatenate the class embedding to the hidden representation. We use the following method to solve this problem. For each testing sample, we concatenate the class embedding of each class c to its hidden representation as its new hidden representation and then compute the prediction probability that the testing sample belongs to class c via the softmax function used in the multi-task neural network. Finally we choose class c with the largest prediction probability as the predicted label. In mathematics, we predict the class label of a testing sample as

$$c^* = \arg\max_{r \in [k]} \mathbb{P}(y = r | f(\mathrm{concat}(\gamma(\mathbf{x}_*^i), \hat{\mathbf{e}}_t^i, \hat{\mathbf{e}}_c^{i,r}))), \quad (4)$$

where $[k]$ denotes a set of positive integers no larger than k and $\mathbf{x}_*^i$ denotes the test data point from the ith task. Note that in the prediction rule (4), the concatenated class embedding $\hat{\mathbf{e}}_c^{i,r}$ changes with r.

3.4 Extension to Regression Tasks

For the regression problems, there are only continuous labels and we cannot define class embeddings. So we only use task embeddings as the augmented feature representation. Furthermore, the adjacency matrix $\mathbf{G}^i$ for the ith task is constructed differently from classification tasks. Specifically, the (p, q)th entry in $\mathbf{G}^i$, g_{pq}^i, for a regression task is defined as

$$g_{pq}^i = \exp\{-\|\hat{\mathbf{h}}_p^i - \hat{\mathbf{h}}_q^i\|_2^2\}.$$

Since there is no class embedding, we do not need the prediction rule as in Eq. (4). The rest is identical to classification tasks.

3.5 Analysis

The proposed approach to augment the feature representation based on HGNN is interesting and here we provide some analyses to give insights into this model.

To understand why the incorporation of the task embedding can improve the performance, we use single-task learning as an example to make an illustration, which also works for multi-task learning. We denote the learning function without the use of the task embedding as $g(\mathbf{x})$ and that with the task embedding as $\hat{g}(\text{concat}(\mathbf{x},\mathbf{e}))$, where $\mathbf{e}$ denotes the task embedding. If those two learning functions are within the same family such as the linear model or the neural network, we can see that $\hat{g}(\text{concat}(\mathbf{x},\mathbf{e}))$ can reduce to $g(\mathbf{x})$ when all the parameters related to $\mathbf{e}$ in $\hat{g}(\cdot)$ are set to 0, making the expressiveness of $g(\mathbf{x})$ lower than that of $\hat{g}(\text{concat}(\mathbf{x},\mathbf{e}))$. Hence, given the same training dataset, the training loss of $\hat{g}(\text{concat}(\mathbf{x},\mathbf{e}))$ is usually lower than that of $g(\mathbf{x})$. Of course, the model complexity of $\hat{g}(\text{concat}(\mathbf{x},\mathbf{e}))$ is larger than that of $g(\mathbf{x})$. Based on generalization bounds derived in single-task learning based on for example the Rademacher complexity [4], the generalization loss is upper-bounded by the sum of the training loss and the model complexity. So if the decrease of the training loss of $\hat{g}(\text{concat}(\mathbf{x},\mathbf{e}))$ compared with $g(\mathbf{x})$ is larger than the increasing of the model complexity in $\hat{g}(\text{concat}(\mathbf{x},\mathbf{e}))$, then $\hat{g}(\text{concat}(\mathbf{x},\mathbf{e}))$ is likely to have a lower generalization loss than $g(\mathbf{x})$. Such analysis also holds for multi-task learning. In experiments, we find that when the dimension of the task embedding is small, leading to a small number of additional parameters incurred as well as a low model complexity, the generalization performance is better than that with a larger dimension for the task embedding, which verifies the above analysis.

The input space, which is a subset of a vector space, is denoted by $\mathcal{X}$ and the output space is denoted by $\mathcal{Y}$. Training samples $\{(\mathbf{x}_i,\mathbf{y}_i)_{i=1}^n\} \in \mathcal{X}\times\mathcal{Y}$ are distributed according to some unknown distribution P. Let $\ell : \mathbb{R}^k \times \mathcal{Y} \to \mathbb{R}^+$ be the loss function, where k denotes the dimension of the label space. The learning function is defined as $f(\mathbf{x}) = \mathbf{W}^\intercal\mathbf{x}$ where the superscript $^\intercal$ denotes the transpose and $\mathbf{W}$ is abused to denote the parameter in this linear learner. The expected loss is defined as $\mathcal{L}(\mathbf{W}) = \mathbb{E}[\ell(\mathbf{W}^\intercal\mathbf{x},\mathbf{y})]$. The empirical loss is defined as $\hat{\mathcal{L}}(\mathbf{W}) = \frac{1}{n}\sum_{i=1}^n \ell(\mathbf{W}^\intercal\mathbf{x}_i,\mathbf{y}_i)$. The data matrix $\mathbf{X}$ is defined as $\mathbf{X} = (\mathbf{x}_1,\ldots,\mathbf{x}_n) \in \mathbb{R}^{p\times n}$ and the label matrix $\mathbf{Y}$ is defined as $\mathbf{Y} = (\mathbf{y}_1,\ldots,\mathbf{y}_n) \in \mathbb{R}^{k\times n}$. $\mathbf{e} \in \mathbb{R}^{q\times 1}$ denotes the task embedding and $\mathbf{E} = \mathbf{e}\mathbf{1}^\intercal \in \mathbb{R}^{q\times n}$ is the task embedding matrix for all the training data, where $\mathbf{1}$ denotes a column vector of all ones with an appropriate size.

Let us consider two models. The objective function of model 1 is formulated as

$$\hat{\mathbf{W}}_1 = \underset{\mathbf{W_1}}{\text{argmin}} \|\mathbf{Y} - \mathbf{W}_1^\intercal\mathbf{X}\|_2^2 + \lambda\|\mathbf{W_1}\|_2^2, \tag{5}$$

and that of model 2 is

$$\hat{\mathbf{W}}_2 = \underset{\mathbf{W_2}}{\text{argmin}} \|\mathbf{Y} - \mathbf{W}_2^\intercal\hat{\mathbf{X}}\|_2^2 + \lambda\|\mathbf{W_2}\|_2^2, \tag{6}$$

where $\mathbf{W_1} \in \mathbb{R}^{p\times k}$, $\mathbf{W_2} \in \mathbb{R}^{(q+p)\times k}$, $\hat{\mathbf{x}}_i = (\mathbf{x}_i^\intercal, \mathbf{e}^\intercal)^\intercal$, $\hat{\mathbf{X}} = (\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_n) = (\mathbf{X}^\intercal,\mathbf{E}^\intercal)^\intercal \in \mathbb{R}^{(q+p)\times n}$. So model 1 is a ridge regression model which can be applied to both classification and regression tasks and model 2 is a variant of model 1 with the task embedding incorporated. For training losses of those two models, we have the following result, whose proof is in the appendix.

Theorem 1. *If* $\mathbf{X}$ *and* $\mathbf{E}$ *satisfy* $\mathbf{X}^\intercal\mathbf{X}\mathbf{E}^\intercal\mathbf{E}+\mathbf{E}^\intercal\mathbf{E}\mathbf{X}^\intercal\mathbf{X}+2\lambda\mathbf{E}^\intercal\mathbf{E}+\mathbf{E}^\intercal\mathbf{E}\mathbf{E}^\intercal\mathbf{E}\succeq 0$ *where* $\mathbf{M}_1 \succeq \mathbf{M}_2$ *means that* $\mathbf{M}_1 - \mathbf{M}_2$ *is positive semidefinite, then the training loss of model 2 with the task embedding is always lower than that of model 1 without the task embedding. That is, we have*

$$\|\mathbf{Y}-\hat{\mathbf{W}}_1^\intercal\mathbf{X}\|_2^2 \geq \|\mathbf{Y}-\hat{\mathbf{W}}_2^\intercal\hat{\mathbf{X}}\|_2^2. \tag{7}$$

Remark 1. Theorem 1 implies that for a model, incorporating the task embedding to augment the feature representation will incur a lower training loss than that without the task embedding. From the perspective of the model capacity, model 1 is a reduced version of model 2 by setting the task embedding to be zero and hence mode 2 has a larger capacity than model 1, making model 2 possess a large chance to have a lower training loss. The condition proposed in Theorem 1 is very easy to check and we can adjust λ to ensure the positive semidefiniteness of the condition.

We also analyze the generalization bound of the two models. We first rewrite problems (5) and (6) into equivalent formulations as

$$\begin{aligned}\hat{\mathbf{W}}_1 &= \operatorname*{argmin}_{\|\mathbf{W_1}\|_2\leq W_*} \|\mathbf{Y}-\mathbf{W}_1^\intercal\mathbf{X}\|_2^2\\ \hat{\mathbf{W}}_2 &= \operatorname*{argmin}_{\|\mathbf{W_2}\|_2\leq W_*} \|\mathbf{Y}-\mathbf{W}_2^\intercal\hat{\mathbf{X}}\|_2^2.\end{aligned}$$

For the above two problems, we have the following result, whose proof is in the appendix.

Theorem 2. *Suppose* $\|\mathbf{x}_i\|, \|\hat{\mathbf{x}}_i\| \leq X_*$, *the task embedding satisfies the condition in Theorem 1. Then for any* $\delta > 0$, *with probability at least* $1-\delta$, *we have*

$$\begin{aligned}\mathbb{E}_{\mathbf{x},\mathbf{y}}(\|\mathbf{y}-\hat{\mathbf{W}}_1^\intercal\mathbf{x}\|_2^2) &\leq \frac{1}{n}\sum_{i=1}^n\|\mathbf{y}_i-\hat{\mathbf{W}}_1^\intercal\mathbf{x}_i\|_2^2 + 4X_*\beta_*\sqrt{\frac{1}{n}}+2X_*\beta_*\sqrt{\frac{\log(1/\delta)}{2n}}\\ \mathbb{E}_{\mathbf{x},\mathbf{y}}(\|\mathbf{y}-\hat{\mathbf{W}}_2^\intercal\hat{\mathbf{x}}\|_2^2) &\leq \frac{1}{n}\sum_{i=1}^n\|\mathbf{y}_i-\hat{\mathbf{W}}_2^\intercal\hat{\mathbf{x}}_i\|_2^2 + 4X_*\beta_*\sqrt{\frac{1}{n}}+2X_*\beta_*\sqrt{\frac{\log(1/\delta)}{2n}}.\end{aligned}$$

Remark 2. According to Theorem 2, the generalization upper-bound of model 2 with the use of the task embedding is lower than that without the task embedding because of the lower training loss of model 2 which has been proved in Theorem 1. This may imply that there is a large chance that the expected loss of model 2 is lower than that of model 1, which can be verified in empirical studies.

4 Experiments

In this section, we conduct empirical studies to test the performance of the proposed HGNN.

4.1 Experimental Settings

We conduct experiments on several benchmark datasets, including **ImageCLEF** [7], **Office-Caltech-10** [10], **Office-Home** [34], and **SARCOS** [39].

The **ImageCLEF** dataset is the benchmark for Image-CLEF domain adaptation challenge which contains about 2,400 images from 12 common categories shared by four tasks including *Caltech-256* (**C**), *ImageNet ILSVRC* (**I**), *Pascal VOC 2012* (**P**), and *Bing* (**B**). There are 50 images in each category and 600 images in each task.

The **Office-Caltech-10** dataset includes 10 common categories shared by the Office-31 and Caltech-256 datasets. It contains four domains: *Caltech* (**C**) that is sampled from Caltech-256 dataset, *Amazon* (**A**) that contains images collected from the amazon website, *Webcam* (**W**) and *DSLR* (**D**) that are taken by the web camera and DSLR camera under the office environment. In our experiment, we regard each domain as a task.

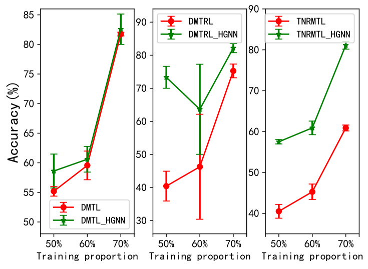
The **Office-Home** dataset has 15,500 images across 65 classes in the office and home settings from four domains with a large domain discrepancy: *Artistic images* (**Ar**), *Clip art* (**Cl**), *Product images* (**Pr**), and *Real-world images* (**Rw**). In our experiment, we regard each domain as a task.

The **SARCOS** dataset studies a multi-output problem of learning the inverse dynamics of 7 SARCOS anthropomorphic robot arms, each of which corresponds to a task, based on 21 features, including seven joint positions, seven joint velocities, and seven joint accelerations. By following [39], we treat each output as a task and randomly sample 2000 data points from each output to construct the dataset.

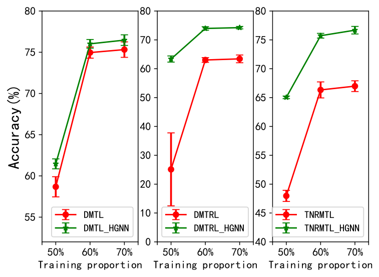
Since the proposed HGNN can be combined with many deep multi-task learning models as discussed before, we incorporate the HGNN into the Deep Multi-Task Learning (DMTL) which shares the first several layers as the common hidden feature representation for all the tasks as did in [8, 18, 22, 27, 37], Deep Multi-Task Representation Learning (DMTRL) [35], and Trace Norm Regularised Deep Multi-Task Learning (TNRMTL) [36], respectively, to show the benefit of the learned augmented features.

In experiments, we use the Tensorflow package [1] to implement all the models and leverage the VGG-19 network [32] pretrained on the ImageNet dataset [29] as the backbone of the feature extractor. After that, all the multi-task learning models adopt a two-layer fully-connected architecture (#data_dim $\times$ 600 $\times$ #classes) and the ReLU activation function is used. The first layer is shared by all tasks to learn a common representation corresponding to Eq. (1) and the second layer is for task-specific outputs.

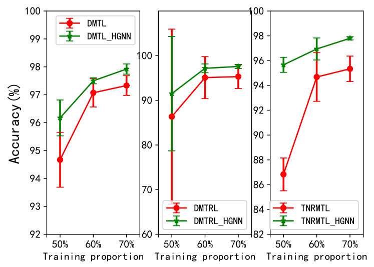
For optimization, we use the Adam method [16] with the learning rate varying as $\eta = \frac{0.02}{1+p}$, where p is the number of the iteration. By following GAT, the dimension of task embeddings is set to 8, i.e., $F'_t = 8$. Similarly, we also set the dimension of class embeddings to 8, i.e., $F'_c = 8$. The size of mini-batch is set to 32. Each experiment repeats for 5 times and we report the average performance as well as the standard deviation.



(a)



(b)



(c)

Fig. 2. Performance of different models on different datasets when varying with the training proportion: (a) ImageCLEF; (b) Office-Home; (c) Office-Caltech-10.

4.2 Experimental results

Results on Classification Tasks For classification tasks, the performance measure is the classification accuracy. To investigate the effect of the size of the training dataset on the performance, we vary the proportion of training data from 50% to 70% at an interval of 10% and plot the average test accuracy of different methods in Figures 2(a)-2(c). According to results reported in these figures, we can see that the incorporation of the HGNN into baseline models improves the classification accuracy of all baseline models especially when the training proportion is small. As reported in Figures 2(b) and 2(c), the incorporation of the HGNN boosts the performance of all the baseline on the Office-Caltech-10 and Office-Home datasets. For the DMTRL and TNRMTL models, the improvement is significant with the use of the HGNN. Moreover, when using augmented features learned by the HGNN, the standard deviation becomes smaller than the corresponding baseline model without using the HGNN under every experimental setting, which implies that the HGNN can improve the stability of baseline models to some extent.

Results on Regression Tasks For regression tasks, we use the mean squared error to measure the performance. The test errors on the SARCOS dataset are shown in Figure 3 where the training proportion varies from 50% to 70% at an interval of 10%. As shown in Figure 3, after using the HGNN, the test error of each baseline model has a significant decrease at each training proportion, which demonstrate the effectiveness of augmented features learned in the HGNN method.

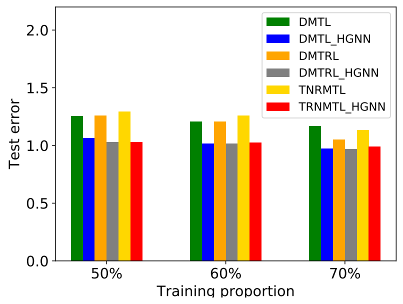


Fig. 3. Performance of different models on the SARCOS dataset.

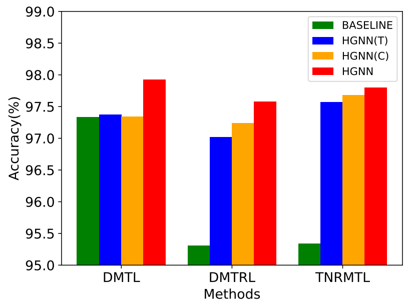


Fig. 4. The ablation study on the Office-Caltech-10 dataset.

4.3 Ablation Study

To study the effectiveness of task embeddings and class embeddings in the HGNN model, we study two variants of HGNN, including HGNN(T) that only augments with the task embedding and HGNN(C) that only augments with the class embedding. The comparison among baseline models, HGNN, variants of HGNN on the Office-Caltech-10 dataset is shown in Figure 4. According to the results, we can see that the use of only the class embedding in HGNN(C) or the task embedding in HGNN(T) can improve the performance over baseline models, which shows that augmented features learned in two ways are effective. HGNN(C) seems better than HGNN(T) in this experiment. One reason is that class embeddings may contain more discriminative features for the classification task. Figure 4 also indicates that using both task embeddings and class embeddings achieves the best performance, which again verifies the usefulness of the HGNN.

4.4 Visualization

To dive deeper into the learned features, we plot in Figures 5 and 6 the t-SNE embeddings of the feature representations learned for the four tasks on the Office-Caltech-10 dataset by TNRMTL and TNRMTL_HGNN, respectively, at the training and testing processes. We observe that the data based on the representation derived by the HGNN model are more separable among classes in each task during either the training process or the testing process. This phenomenon verifies the effectiveness of the augmented features learned in the HGNN to help discriminate data points in different classes of all the tasks.

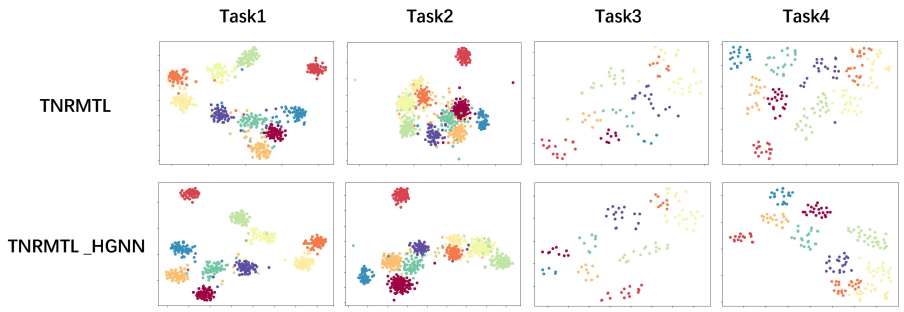


Fig. 5. The feature visualization by the t-SNE method for the training data in the four tasks on the Office-Caltech-10 dataset. Different markers and different colors are used to denote different classes. (Best viewed in color.)

4.5 Sensitivity Analysis

We conduct the sensitivity analysis of the performance with respect to the dimension of task embedding (denoted by F'_t) and class embedding (denoted by F'_c), respectively, on the ImageCLEF dataset. The results are shown in Tables 1 and 2. According to the results, we can see that $F'_t = 8$ and $F'_c = 8$ are a good choice in most cases, though in some case, a lower value (i.e., 4) performs better. When the dimension is not so large (e.g., not large than 32), the performance changes a little, making the choice of the dimension insensitive. However, when using a larger dimension (e.g., 64), the classification accuracy drops significantly, implying that the HGNN prefers a small dimension.

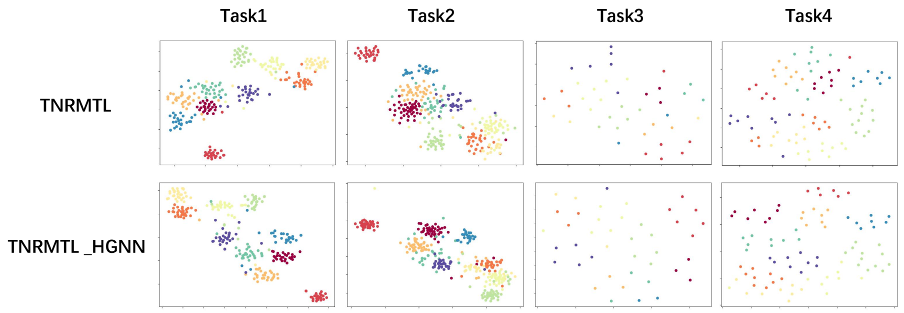


Fig. 6. The feature visualization by the t-SNE method for the testing data in the four tasks on the Office-Caltech-10 dataset. Different markers and different colors are used to denote different classes. (Best viewed in color.)

Table 1. The classification accuracy (%) on ImageCLEF when varying F'_t and fixing F'_c as 8.

F'_t	4	8	16	32	64
DMTL_HGNN	81.22±1.27	**82.03**±1.98	81.18±1.43	80.41±0.88	79.89±0.65
DMTRL_HGNN	81.21±0.80	**82.07**±1.47	81.64±0.77	81.15±2.35	81.48±1.78
TNRMTL_HGNN	**82.35**±1.66	81.11±0.94	82.27±1.19	81.74±0.91	81.20±0.40

Table 2. The classification accuracy (%) on ImageCLEF when varying F'_c and fixing F'_t as 8.

F'_c	4	8	16	32	64
DMTL_HGNN	**82.37**±1.13	82.03±1.98	81.75±0.56	80.71±1.30	81.65±2.84
DMTRL_HGNN	81.73±1.39	**82.07**±1.47	80.40±1.57	81.94±2.03	80.74±2.16
TNRMTL_HGNN	80.07±3.36	**81.11**±0.94	80.64±1.11	80.17±0.98	80.47±0.80

5 Conclusion

In this paper, we propose a hierarchical graph neural network to learn augmented features for deep multi-task learning. The proposed HGNN has two levels. In the first level, the intra-task graph neural network is used to learn a powerful representation for each data point in a task by aggregating information from its neighbors in this task. Based on the learned representation, we can learn the task embedding for each task as well as the class embedding if any. The inter-task graph neural network as well inter-class graph neural network is used to update each task embedding and each class embedding. Finally the learned task embedding and class embedding can be used to augment the data representation. Extensive experiments show the effectiveness of the proposed HGNN. In our future work, we are interested in applying the HGNN to other multi-task learning models.

Acknowledgements

This work is supported by NSFC grant 62076118.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation. pp. 265–283 (2016)
2. Ando, R.K., Zhang, T.: A framework for learning predictive structures from multiple tasks and unlabeled data. Journal of Machine Learning Research **6**, 1817–1853 (2005)

3. Argyriou, A., Evgeniou, T., Pontil, M.: Multi-task feature learning. In: Advances in Neural Information Processing Systems 19. pp. 41–48 (2006)
4. Bartlett, P.L., Mendelson, S.: Rademacher and Gaussian complexities: Risk bounds and structural results. Journal of Machine Learning Research **3**, 463–482 (2002)
5. Bickel, S., Bogojeska, J., Lengauer, T., Scheffer, T.: Multi-task learning for HIV therapy screening. In: Proceedings of the Twenty-Fifth International Conference on Machine Learning. pp. 56–63 (2008)
6. Bonilla, E., Chai, K.M.A., Williams, C.: Multi-task Gaussian process prediction. In: Advances in Neural Information Processing Systems 20. pp. 153–160. Vancouver, British Columbia, Canada (2007)
7. Caputo, B., Müller, H., Martinez-Gomez, J., Villegas, M., Acar, B., Patricia, N., Marvasti, N., Üsküdarlı, S., Paredes, R., Cazorla, M., et al.: ImageCLEF 2014: Overview and analysis of the results. In: International Conference of the Cross-Language Evaluation Forum for European Languages. pp. 192–211. Springer (2014)
8. Caruana, R.: Multitask learning. Machine Learning **28**(1), 41–75 (1997)
9. Gao, Y., Bai, H., Jie, Z., Ma, J., Jia, K., Liu, W.: MTL-NAS: Task-agnostic neural architecture search towards general-purpose multi-task learning. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2020)
10. Gong, B., Shi, Y., Sha, F., Grauman, K.: Geodesic flow kernel for unsupervised domain adaptation. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2066–2073. IEEE (2012)
11. Han, L., Zhang, Y.: Learning multi-level task groups in multi-task learning. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence (2015)
12. Han, L., Zhang, Y.: Multi-stage multi-task learning with reduced rank. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence (2016)
13. Jacob, L., Bach, F., Vert, J.P.: Clustered multi-task learning: A convex formulation. In: Advances in Neural Information Processing Systems 21. pp. 745–752 (2008)
14. Jalali, A., Ravikumar, P.D., Sanghavi, S., Ruan, C.: A dirty model for multi-task learning. In: Advances in Neural Information Processing Systems 23. pp. 964–972. Vancouver, British Columbia, Canada (2010)
15. Kim, R., So, C.H., Jeong, M., Lee, S., Kim, J., Kang, J.: HATS: A hierarchical graph attention network for stock movement prediction. CoRR **abs/1908.07999** (2019)
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
17. Kumar, A., III, H.D.: Learning task grouping and overlap in multi-task learning. In: Proceedings of the 29 th International Conference on Machine Learning. Edinburgh, Scotland, UK (2012)
18. Li, S., Liu, Z., Chan, A.B.: Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network. IJCV **113**(1), 19–36 (2015)
19. Liu, H., Palatucci, M., Zhang, J.: Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In: Proceedings of the 26th Annual International Conference on Machine Learning (2009)
20. Liu, P., Fu, J., Dong, Y., Qiu, X., Cheung, J.C.K.: Multi-task learning over graph structures. CoRR **abs/1811.10211** (2018)
21. Liu, S., Johns, E., Davison, A.J.: End-to-end multi-task learning with attention. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. pp. 1871–1880 (2019)
22. Liu, W., Mei, T., Zhang, Y., Che, C., Luo, J.: Multi-task deep visual-semantic embedding for video thumbnail selection. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. pp. 3707–3715 (2015)

23. Lozano, A.C., Swirszcz, G.: Multi-level lasso for sparse multi-task regression. In: Proceedings of the 29th International Conference on Machine Learning. Edinburgh, Scotland, UK (2012)
24. Lu, H., Huang, S.H., Ye, T., Guo, X.: Graph star net for generalized multi-task learning. CoRR **abs/1906.12330** (2019)
25. Meng, Z., Adluru, N., Kim, H.J., Fung, G., Singh, V.: Efficient relative attribute learning using graph neural networks. In: Proceedings of the 15th European Conference on Computer Vision (2018)
26. Misra, I., Shrivastava, A., Gupta, A., Hebert, M.: Cross-stitch networks for multi-task learning. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. pp. 3994–4003 (2016)
27. Mrksic, N., Séaghdha, D.Ó., Thomson, B., Gasic, M., Su, P., Vandyke, D., Wen, T., Young, S.J.: Multi-domain dialog state tracking using recurrent neural networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics. pp. 794–799 (2015)
28. Obozinski, G., Taskar, B., Jordan, M.: Multi-task feature selection. Tech. rep., Department of Statistics, University of California, Berkeley (June 2006)
29. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**(3), 211–252 (2015)
30. Ryu, H., Shin, H., Park, J.: Multi-agent actor-critic with hierarchical graph attention network. CoRR **abs/1909.12557** (2019)
31. Shinohara, Y.: Adversarial multi-task learning of deep neural networks for robust speech recognition. In: Proceedings of the 17th Annual Conference of the International Speech Communication Association. pp. 2369–2372 (2016)
32. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
33. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
34. Venkateswara, H., Eusebio, J., Chakraborty, S., Panchanathan, S.: Deep hashing network for unsupervised domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5018–5027 (2017)
35. Yang, Y., Hospedales, T.M.: Deep multi-task representation learning: A tensor factorisation approach. In: Proceedings of the 6th International Conference on Learning Representations (2017)
36. Yang, Y., Hospedales, T.M.: Trace norm regularised deep multi-task learning. In: Proceedings of the 6th International Conference on Learning Representations, Workshop Track (2017)
37. Zhang, W., Li, R., Zeng, T., Sun, Q., Kumar, S., Ye, J., Ji, S.: Deep model based transfer and multi-task learning for biological image analysis. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1475–1484 (2015)
38. Zhang, Y., Yang, Q.: A survey on multi-task learning. CoRR **abs/1707.08114** (2017)
39. Zhang, Y., Yeung, D.Y.: A convex formulation for learning task relationships in multi-task learning. In: Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence. pp. 733–742 (2010)
40. Zhang, Y., Wei, Y., Yang, Q.: Learning to multitask. In: Advances in Neural Information Processing Systems 31. pp. 5776–5787 (2018)
41. Zhang, Y., Yeung, D., Xu, Q.: Probabilistic multi-task feature selection. In: Advances in Neural Information Processing Systems 23. pp. 2559–2567 (2010)