Multiple Relational Attention Network for Multi-task Learning

Jiejie Zhao[1], Bowen Du[1*], Leilei Sun[1], Fuzhen Zhuang[2,3], Weifeng Lv[1], Hui Xiong[4]
[1]SKLSDE and BDBC Lab, Beihang University, Beijing 100083, China,
[2]Key Lab of IIP of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China,
[3]University of Chinese Academy of Sciences, Beijing 100049, China,
[4]Department of Management Science and Information Systems, Rutgers University
[1]{zjj,dubowen,leileisun,lwf}@buaa.edu.cn, [2,3]zhuangfuzhen@ict.ac.cn, [4]hxiong@rutgers.edu

ABSTRACT

Multi-task learning is a successful machine learning framework which improves the performance of prediction models by leveraging knowledge among tasks, e.g., the relationships between different tasks. Most of existing multi-task learning methods focus on guiding learning process by predefined task relationships. In fact, these methods have not fully exploited the associated relationships during the learning process. On the one hand, replacing predefined task relationships by adaptively learned ones may result in higher prediction accuracy as it can avoid the risk of misguiding caused by improperly predefined relationships. On the other hand, apart from the task relationships, feature-task dependence and feature-feature interactions could also be employed to guide the learning process. Along this line, we propose a *M*ultiple *R*elational *A*ttention *N*etwork (MRAN) framework for multi-task learning, in which three types of relationships are considered. Correspondingly, MRAN consists of three attention-based relationship learning modules: 1) a task-task relationship learning module which captures the relationships among tasks automatically and controls the positive and negative knowledge transfer adaptively; 2) a feature-feature interaction learning module that handles the complicated interactions among features; 3) a task-feature dependence learning module, which can associate the related features with target tasks separately. To evaluate the effectiveness of the proposed MARN, experiments are conducted on two public datasets and a real-world dataset crawled from a review hosting site. Experimental results demonstrate the superiority of our method over both classical and the state-of-the-art multi-task learning methods.

CCS CONCEPTS

• **Computing methodologies** → **Multi-task learning**; **Neural networks**; • **Information systems** → *Data mining*.

KEYWORDS

Multi-task Learning, Task Relationship, Relationship Learning, Attention

ACM Reference Format:

Jiejie Zhao, Bowen Du, Leilei Sun, Fuzhen Zhuang, Weifeng Lv, Hui Xiong. 2019. Multiple Relational Attention Network for Multi-task Learning. In *The 25th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD'19), August 4–8, 2019, Anchorage, AK, USA.* ACM, NY, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330861

1 INTRODUCTION

Multi-task learning (MTL) aims to improve the generalization ability of prediction models by utilizing general knowledge contained in multiple tasks [3]. It has been successfully applied in a wide range of areas such as computer vision [21], natural language processing [23], and information retrieval [20]. However, the crucial problems in MTL are how to model relationships among tasks appropriately and how to exploit them to improve the performance of each task.

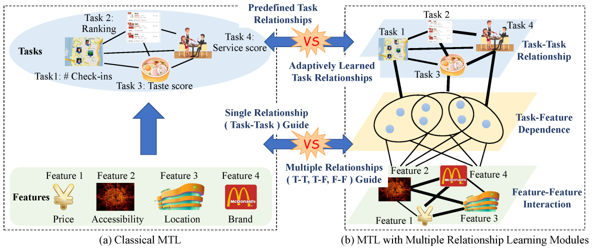
Generally, the existing MTL approaches can be classified into two categories: regularization-based MTL and deep learning based MTL (DL-MTL). Most of the regularization-based MTL methods were designed under the LASSO (Least Absolute Shrinkage and Selection Operator) framework [1, 31]. On the one hand, users need to define the task relationships beforehand, and there may exist negative knowledge transfer if the task relationships are not properly defined. On the other hand, LASSO cannot provide enough representation capacity as it is linear and shallow essentially. Recent efforts have focused on DL-MTL methods. Some of them study how to improve the prediction accuracy by enhancing the representation capacity with deeper learning architectures [2, 17], while the others explore how to determine task relationships by models themselves [7, 17].

In our opinion, the performance of MTL methods could be further improved in the following two directions: 1) *Replacing the predefined task relationships by adaptively learned ones.* Since an improper task relationship may result in negative knowledge transfer [10], it is essential for a MTL method to be equipped with an adaptive relationship learning module. 2) *Integrating multiple relationships to guide the learning process.* Apart from task-task relationships which have been widely studied in existing research, task-feature dependence and feature-feature interactions could also be leveraged to guide the learning process of MTL. Figure 1 demonstrates the differences between MRAN and the classical MTL. The two ideas have great potential to bring the existing MTL research to a new height, yet they are still faced with the following challenges.

The first difficulty is *how to learn the task-task relationships adaptively by MTL model itself.* Generally, the relationships among tasks could be positively related, negatively related, or unrelated. The correlation of tasks could be either strong or weak. The tasks may be linearly unrelated, but non-linearly related. A task-task relationship

∗ Corresponding Author.

KDD '19, August 4–8, 2019, Anchorage, AK, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330861



(a) Classical MTL

(b) MTL with Multiple Relationship Learning Modules

Figure 1: Classical multi-task learning method versus multiple relational attention network.

learning module must be able to capture the task-task relationships from the following three perspectives: transfer polarity, correlation strength, and association manner. Moreover, how to effectively exploit the discovered task-task relationships to improve the learning process is also required to be studied. At present, rare research has been done to solve this problem[16, 17], and they can handle the adaptive relationships of less amount of tasks (around two or three).

The second difficulty is *how to associate the different tasks with related features*. In many practical applications, a task may be related with various features; different tasks may rely on different subsets of all the studied features; the feature subsets of different tasks may also be overlapped with each other. For example, the popularity of a store may rely on accessibility features such as bike stations around the store; the rating of the store may rely on features of service quality; the ranking of the store may rely on both the two types of features. Additionally, it is also challenging to model the interactions of features. Though this problem has been widely studied in previous research under the name of feature engineering, most of the previous research are not suitable for multi-output prediction. What’s more, designing a comprehensive framework to incorporate the multiple relationships simultaneously, and to exploit the multiple relationships effectively is also challenging.

To fill this research gap, we design a *M*ultiple *R*elational *A*ttention *N*etwork (MRAN) for multi-task learning in this paper. Different from the existing MTL methods, our approach leverages multiple self-learned relationships simultaneously to improve the prediction accuracy of each task. In particular, 1) we first propose a self-attention based relationship learning module to learn relationships among tasks adaptively. This module could not only capture pairwise task relationships via task correlation attention matrix but also achieve an intrinsic hidden representation of tasks. 2) A similar self-attention mechanism is provided to model the interactions of features and reduce the redundancy among them. This module can help us obtain a better feature representation for all tasks. 3) An alignment attention module is designed to specify the shared features for different tasks. This module is used to construct different feature representations for different tasks separately. Finally, a deep multi-task learning framework equipped with three aforementioned modules is proposed to predict the multiple outputs.

In summary, this paper has the following contributions:

- *An adaptive task-task relationship learning module for MTL method*, which is able to capture the relationships between tasks automatically and then control the positive and negative knowledge transfer among tasks accordingly.
- *A task-feature alignment module*, which specifies the shared features for different tasks separately. This module is able to extract a customized representation of shared features for each target task severally.
- *A comprehensive deep MTL learning framework equipped with multiple relationship learning modules*, which improves the prediction accuracy of each task by leveraging task-task knowledge transfer, task-feature alignment, and feature-feature interaction simultaneously.

2 PRELIMINARIES

In this section, we first present the definition of multi-task problems, then introduce the formalization of regularization-based MTL and DL-MTL separately.

2.1 Definition

According to literature [30], multi-task learning is defined as:

Definition 1 Multi-Task Learning Given m learning tasks $\left\{\mathcal{T}^t\right\}_{t=1}^m$, where all the tasks or a subset of them are related, *multi-task learning* aims to help improve the learning performance of a model for $\mathcal{T}^t$ by leveraging knowledge contained among tasks.

In supervised MTL problems, a task $\mathcal{T}^t$ is accompanied by a training dataset $\mathcal{D}^t$ consisting of features $\{\mathbf{x}_i^t\}_{i=1}^{n^t}$ and labels $\{y_i^t\}_{i=1}^{n_t}$, where $\mathbf{x}_i^t \in \mathbb{R}^{1\times d}$ and $y_i^t \in \mathbb{R}$. Multi-label learning or multi-output prediction is one of the most popular MTL types, which assumes that different tasks have the same instances. This paper focuses on this type of MTL, thus the dataset contains n data instances $\{\mathbf{x}_i\}_{i=1}^n$ as well as their labels $\{\mathbf{y}_i\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^{1\times d}$ and $\mathbf{y}_i = \{y_i^1, \cdots, y_i^t, \cdots, y_i^m\} \in \mathbb{R}^{1\times m}$.

Our goal is to build a deep network to predict multiple outputs of tasks simultaneously, in which we will exploit T-T transfer relationships, T-F dependence relationships, and F-F interaction relationships to bridge different tasks effectively and robustly.

2.2 Regularization-based Multi-task Learning

In linear MTL models, for a data instance $\mathbf{x}_i \in \mathbb{R}^{1\times d}$ and its label $y_i^t \in \mathbb{R}$, the following equation is used to predict the label:

$$y_i^t = \mathbf{x}_i \mathbf{w}^t + \epsilon, \tag{1}$$

where $\mathbf{w}^t \in \mathbb{R}^d$ is a weight vector that needs to be estimated during the learning process, and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian noise.

To capture the feature interactions, Equation (2) is generally used to model the linear interaction of features [13],

$$y_i^t = \mathbf{x}_i \mathbf{w}^t + \sum_{j=1}^{d} \sum_{k=1}^{d} x_{i,j} x_{i,k} \mathrm{Q}_{j,k}^t + \epsilon, \tag{2}$$

where the interactions of features are represented by a tensor $\mathrm{Q} \in \mathbb{R}^{d\times d\times m}$, $(x_{i,j} x_{i,k} \mathrm{Q}_{j,k}^t)$ describes the interaction of j-th feature with k-th feature with regard to task $\mathcal{T}^t$. The loss function is formulated as follows:

$$\min_{\mathbf{W},\mathrm{Q}} \mathcal{L}(\mathbf{W}, \mathrm{Q}; \mathbf{X}, \mathbf{Y}) + \Omega_F(\mathbf{W}) + \Omega_I(\mathrm{Q}), \tag{3}$$

where $\mathbf{W} = [\mathbf{w}^1 \cdots \mathbf{w}^m] \in \mathbb{R}^{d\times m}$, regularization $\Omega_F(\mathbf{W})$ provides task relatedness in the original feature space, while regularization $\Omega_I(\mathrm{Q})$ encodes the knowledge about how feature interactions are related among tasks.

To capture task relationships, many MTL models are based on the assumption that each task parameters are generated by linearly combining the parameters of relevant tasks [12]. In this case, the parameter matrix $\mathbf{W}$ can be decomposed as $\mathbf{W} = \mathbf{LS}$. Here, $\mathbf{L} \in \mathbb{R}^{d\times k}$ is the collection of k latent bases, while $\mathrm{S} \in \mathbb{R}^{k\times m}$ is the coefficient matrix for linearly combining those bases. The objective function is formulated as:

$$\min_{\mathbf{L},\mathrm{S}} \sum_{t=1}^{m} \mathcal{L}(\mathbf{L}\mathbf{s}^t; \mathrm{X}, \mathrm{Y}^t) + \Omega(\mathbf{L}, \mathrm{S}), \tag{4}$$

where $\mathbf{s}^t$ is t-th column of S to represent $\mathbf{w}^t$ as the linear combination of shared bases $\mathbf{L}$, namely, $\mathbf{w}^t = \mathbf{L}\mathbf{s}^t$, and $\mathrm{Y}^t = \{y_1^t, \ldots, y_n^t\}$.

2.3 Deep Learning based Multi-task Learning

A critical drawback of Equation (1) is that it takes the risk of underfitting since the model relies on the linear feature learning approach which lacks the capacity. To address the drawback, DL-MTL is typically proposed with hard parameter sharing approach. It is generally applied by sharing the L_b bottom layers between all tasks while splitting the L_h top layers for specific tasks. The Shared-bottom network model can be represented as:

$$\begin{aligned} \mathrm{H} &= \sigma(\sigma(\cdots\sigma(\sigma(\mathbf{X}\mathbf{W}^{(1)})\mathbf{W}^{(2)})\cdots\mathbf{W}^{(L_b-1)})\mathbf{W}^{(L_b)}) \\ \mathrm{Y}^t &= (((\mathbf{H}\mathbf{W}^{(L_b+1,t)})\cdots\mathbf{W}^{(L_b+L_h-1,t)})\mathbf{W}^{(L_b+L_h,t)}), \end{aligned} \tag{5}$$

where $\mathbf{W}^{(l)}$ is the weight matrix for the l-th layer and $\sigma(\cdot)$ denotes the non-linearity activation function.

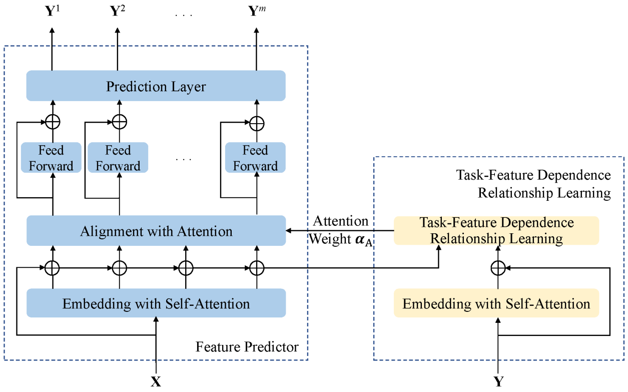
Different from these DL-MTL methods, cross-stitch network is similar to task relationship learning approach of Equation (4) which aims to learn task relationships in terms of the hidden feature representation. Given two tasks A and B with an identical network architecture where $x_{i,j}^{\mathrm{A}}, x_{i,j}^{\mathrm{B}}$ denote the hidden feature in the j-th unit of the i-th hidden layer for task A and B respectively. The cross-stitch operation on $x_{i,j}^{\mathrm{A}}$ and $x_{i,j}^{\mathrm{B}}$ as:

$$\begin{bmatrix} \hat{x}_{i,j}^{\mathrm{A}} \\ \hat{x}_{i,j}^{\mathrm{B}} \end{bmatrix} = \begin{bmatrix} \alpha^{\mathrm{AA}} & \alpha^{\mathrm{AB}} \\ \alpha^{\mathrm{BA}} & \alpha^{\mathrm{BB}} \end{bmatrix} \begin{bmatrix} x_{i,j}^{\mathrm{A}} \\ x_{i,j}^{\mathrm{B}} \end{bmatrix}, \tag{6}$$

where $\hat{x}_{i,j}^{\mathrm{A}}$ and $\hat{x}_{i,j}^{\mathrm{B}}$ are new hidden features after learning two tasks jointly. Here, matrix $\boldsymbol{\alpha} = \begin{bmatrix} \alpha^{\mathrm{AA}} & \alpha^{\mathrm{AB}} \\ \alpha^{\mathrm{BA}} & \alpha^{\mathrm{BB}} \end{bmatrix}$ encodes the relationships between two tasks. However, Equation (6) is on severe negative transfer from unrelated tasks since all feature models equally contribute to different tasks.

3 METHODOLOGY

This section first presents the framework of MRAN in a nutshell, then introduces three relationship modules for MTL one by one. Finally, a theoretical analysis of the proposed approaches is provided.



Y¹
Y²
. . .
Yᵐ
Prediction Layer
Feed Forward
Feed Forward
. . .
Feed Forward
Task-Feature Dependence Relationship Learning
Attention Weight $\boldsymbol{\alpha}_{\mathrm{A}}$
Alignment with Attention
Task-Feature Dependence Relationship Learning
Embedding with Self-Attention
Embedding with Self-Attention
Feature Predictor
X
Y

Figure 2: Overview of the proposed MRAN.

3.1 Framework Overview

This paper proposes a deep multi-task learning framework, which consists of three relationship learning modules for task-task knowledge transfer, task-feature dependence, and feature-feature interaction separately. These modules aim to learn relationships in terms of new representation. Specifically, given input $\mathbf{X}$ and output $\mathbf{Y}$, T-T transfer relationship learning module learns a new task embedding. F-F interaction relationship learning module is similar. Then, T-F dependence relationship learning module produces an attention weight vector $\boldsymbol{\alpha}_{\mathrm{A}}$ based on new feature and task embedding which shows the dependence relationships between tasks and features. During the testing phase, feature predictor network directly performs multiple prediction tasks based on input $\mathbf{X}'$ and the learned attention weight. The T-F dependence relationship learning module is discarded during the testing phase. The overall model architecture is illustrated in Figure 2.

3.2 Task-Task Transfer Relationship Learning with Self-Attention

We exploit four types of self-attention mechanisms to learn task embedding regarding T-T transfer relationship. Given an output unit $\mathrm{Y} \in \mathbb{R}^{n\times m}$, we need first to model transferable relationships between each pair of tasks from the output via four types of self-attention mechanisms as follows:

- *Dot-product Attention.* A dot-product based attention function is to calculate the affinity matrix solely from the output Y as follows:

$$\mathrm{S_O} = \mathrm{Y}^\top\mathrm{Y}. \tag{7}$$

- *General Attention.* An easy way to capture the relationship between each $\mathrm{Y}^t \in \mathrm{Y}$ is using a matrix $\mathbf{W}_{\mathrm{S_O}} \in \mathbb{R}^{n\times n}$, and calculating the affinity matrix as:

$$\mathrm{S_O} = \mathrm{Y}^\top\mathbf{W}_{\mathrm{S_O}}\mathrm{Y}. \tag{8}$$

- *Concatenation-based Attention.* This method first concatenates the output Y and itself, and then the affinity matrix can be obtained via multiplying a weight matrix $\mathbf{W}_{\mathrm{S_O}} \in \mathbb{R}^{q\times 2n}$ by the concatenation, where q is the latent dimensionality. We select *tanh* as the activation function.

$$\mathrm{S_O} = \mathbf{v}_{\mathrm{S_O}}^\top \tanh(\mathbf{W}_{\mathrm{S_O}}[\mathrm{Y};\mathrm{Y}]), \tag{9}$$

where $\mathbf{v}_{S_O} \in \mathbb{R}^{q \times m}$ is the parameter to be learned.

Secondly, in order to learn a single attentive representation, we use the sum-pooling operation to embed the interaction weight matrix, which adds up the value over the corresponding dimension of the feature matrix. The attention weight vector is computed based on the following equation:

$$\boldsymbol{\alpha}_O = \text{softmax}(sum_{col}(S_O)). \tag{10}$$

The standard softmax function aims to convert the input vector into a probability distribution.

Next, the new task representation is computed based on the following equation:

$$\mathrm{H}_O = \boldsymbol{\alpha}_O \odot \mathrm{Y}, \tag{11}$$

where $\odot$ is the Hadamard product.

- *Multi-head Attention.* Another way to calculate the transferable relationship is using multi-head attention mechanism. Figure 3 illustrates the proposed self-attention module using multi-head attention mechanism in our proposed MRAN. Compared with the standard additive attention mechanism which is implemented by using a one-layer feed-forward neural network, multi-head attention mechanism has the ability to jointly attend to information from different representation subspaces in different feature sets.

Given the output matrix Y, multi-head attention mechanism first maps Y to queries (Q), keys (K), and values (V) matrices by using different linear projections. Then h parallel heads are employed to focus on different parts of the output matrix. Formally, for the i-th head, we denote the learned linear maps by $\mathbf{W}_i^{\mathrm{Q}} \in \mathbb{R}^{m/h \times m/h}$, $\mathbf{W}_i^{\mathrm{K}} \in \mathbb{R}^{m/h \times m/h}$ and $\mathbf{W}_i^{\mathrm{V}} \in \mathbb{R}^{m/h \times m/h}$, which correspond to the parameter of queries, keys and values respectively. The affinity weight is calculated as follows:

$$S_{O,i} = \frac{(\mathbf{Q}_i \mathbf{W}_i^{\mathrm{Q}})^\top (\mathbf{K}_i \mathbf{W}_i^{\mathrm{K}})}{\sqrt{m/h}}, \tag{12}$$

where $\mathbf{Q}_i \in \mathbb{R}^{n \times m/h}$ and $\mathbf{K}_i \in \mathbb{R}^{n \times m/h}$. The output is an $m/h \times m/h$ affinity matrix which indicates the similarity among m/h tasks.

Secondly, the attention weight vector is computed based on the following equation:

$$\boldsymbol{\alpha}_{O,i} = \text{softmax}(sum_{col}(S_{O,i})). \tag{13}$$

Thirdly, the new task representations of i-th head is computed based on the following equation:

$$\mathrm{H}_{O,i} = \boldsymbol{\alpha}_{O,i} \odot (\mathbf{V}_i \mathbf{W}_i^{\mathrm{V}}), \tag{14}$$

where $\mathbf{V}_i \in \mathbb{R}^{n \times m/h}$.

Next, all the matrices produced by parallel heads are concatenated together to form a unifying matrix.

$$\mathrm{H}_O = \text{concat}(\mathrm{H}_{O,1}, \cdots, \mathrm{H}_{O,h}). \tag{15}$$

Finally, a residual and batch normalization layer [9] is added in MRAN, that is,

$$\widehat{\mathrm{H}}_O = \text{batchnorm}(\mathrm{H}_O + \mathrm{Y}). \tag{16}$$

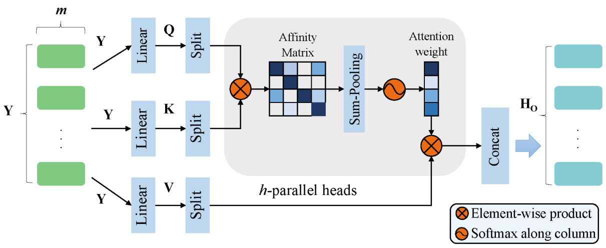


Figure 3: Illustration of the self-attention module.

3.3 Feature-Feature Interaction Relationship Learning with Self-Attention

We assume that feature interactions are common. In order to learn generic feature representations among tasks regarding interaction relationships among features, we need to model the interactions between each pair of features of the input via four types of self-attention mechanisms. Detailed computation is similar to Section 3.2. Here, we take dot-product attention mechanism as an example.

Given an input unit $\mathbf{X} \in \mathbb{R}^{n \times d}$, we first employ four self-attention mechanisms to compute the affinity matrix S_I which indicates the relevance between each feature from $\mathbf{X}$, and we can obtain the affinity matrix S_I with Equation (7). Secondly, we also use sum-pooling operation and softmax function to embed the attention vector $\boldsymbol{\alpha}_I$, and the attention vector can be obtained using Equation (10). Next, we can similarly derive the new feature embedding matrix for input $\mathbf{X}$ as H_I via Equation (11). Finally, a residual and batch normalization layer as in [9] is applied to capturing better feature embedding $\widehat{\mathrm{H}}_I$ which is computed as Equation (16).

3.4 Task-Feature Dependence Relationship Learning with Alignment Attention

We consider the setting that each task may rely on different parts of the shared feature sets. Moreover, we consider not only task relationships in original feature space but also feature interactions. Therefore, it is necessary to model dependence relationships between tasks and features. After modeling T-T transfer relationships and F-F interaction relationships, we use alignment attention to align each task and features to capture task-specific representations (as shown in Figure 4).

3.4.1 *Training Phase.*

In training phase, we employ four types of alignment attention mechanisms to capture the dependence relationships between different tasks and features. Here, we take dot-product attention mechanism as an example, and the other mechanisms are similar.

Given each task embedding matrix $\widehat{\mathrm{H}}_O^t \in \mathbb{R}^n$ (the t-th column of $\widehat{\mathrm{H}}_O$) and feature embedding matrix $\widehat{\mathrm{H}}_I$, we first capture the affinity matrix S_A^t between $\widehat{\mathrm{H}}_O^t$ and $\widehat{\mathrm{H}}_I$ from Equation (7). In the further step, alignment weight for task $\mathcal{T}^t$ is computed as:

$$\boldsymbol{\alpha}_A^t = \text{softmax}(S_A^t). \tag{17}$$

Finally, the new feature representation $\widehat{\mathrm{H}}_A^t$ for task $\mathcal{T}^t$ with $\widehat{\mathrm{H}}_I$ and $\boldsymbol{\alpha}_A^t$ is computed based on Equation (11) and Equation (16).

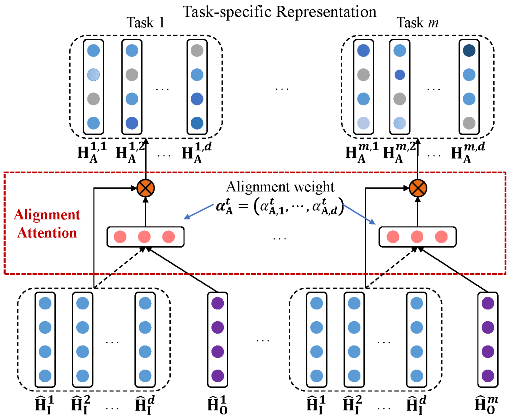

Task 1
Task-specific Representation
Task m
$\mathrm{H}_\mathrm{A}^{1,1}$ $\mathrm{H}_\mathrm{A}^{1,2}$ … $\mathrm{H}_\mathrm{A}^{1,d}$
$\mathrm{H}_\mathrm{A}^{m,1}$ $\mathrm{H}_\mathrm{A}^{m,2}$ … $\mathrm{H}_\mathrm{A}^{m,d}$
Alignment Attention
Alignment weight
$\alpha_\mathrm{A}^t = (\alpha_{\mathrm{A},1}^t, \cdots, \alpha_{\mathrm{A},d}^t)$
$\widehat{\mathrm{H}}_\mathrm{I}^1$ $\widehat{\mathrm{H}}_\mathrm{I}^2$ … $\widehat{\mathrm{H}}_\mathrm{I}^d$ $\widehat{\mathrm{H}}_\mathrm{O}^1$
$\widehat{\mathrm{H}}_\mathrm{I}^1$ $\widehat{\mathrm{H}}_\mathrm{I}^2$ … $\widehat{\mathrm{H}}_\mathrm{I}^d$ $\widehat{\mathrm{H}}_\mathrm{O}^m$

Figure 4: Illustration of alignment attention module. This module computes alignment weight between feature embedding $\widehat{\mathrm{H}}_\mathrm{I} = [\widehat{\mathrm{H}}_\mathrm{I}^1 \cdots \widehat{\mathrm{H}}_\mathrm{I}^d]$ and each task embedding $\widehat{\mathrm{H}}_\mathrm{O}^t$, then computes task-specific representations $\widehat{\mathrm{H}}_\mathrm{A}^t = [\widehat{\mathrm{H}}_\mathrm{A}^{t,1} \cdots \widehat{\mathrm{H}}_\mathrm{A}^{t,d}]$.

3.4.2 Testing Phase. In testing phase, we directly capture task-specific representation by the learned alignment weight for each task. For task $\mathcal{T}^t$, given alignment weight $\boldsymbol{\alpha}_\mathrm{A}^t$ and feature embedding matrix $\widehat{\mathrm{H}}_\mathrm{I}$, task-specific representation $\widehat{\mathrm{H}}_\mathrm{A}^t$ for task t is computed based on Equation (16).

3.5 Prediction Layer

For each task, we add a feed-forward sub-layer to convert the task-specific representation $\widehat{\mathrm{H}}_\mathrm{A}^t$ to final predict result. The feed-forward sub-layer consists of two linear layers with ReLU nonlinearity [18] in the middle. Formally, we have the following equation:

$$\widehat{\mathrm{Y}}^t = \mathrm{ReLU}(\widehat{\mathrm{H}}_\mathrm{A}^t \mathbf{W}_{\mathrm{P}_1}^t)\mathbf{W}_{\mathrm{P}_2}^t, \tag{18}$$

where $\mathbf{W}_{\mathrm{P}_1}^t \in \mathbb{R}^{d \times h_f}$ and $\mathbf{W}_{\mathrm{P}_2}^t \in \mathbb{R}^{h_f}$ are trainable matrices, and $\widehat{\mathrm{Y}}^t$ denotes the prediction result of task $\mathcal{T}^t$.

3.6 Objective Function

To train MRAN, we minimize a loss function $\mathcal{L}_{tot}$ with input $\mathbf{X}$ and task-specific labels $\mathbf{Y}^t \in \mathbf{Y}$, $t = 1, 2, \cdots, m$, which is defined as:

$$\mathcal{L}_{tot}(\mathrm{X}, \mathrm{Y}^{1:m}) = \sum_{t=1}^{m} \lambda^t \mathcal{L}^t(\mathrm{X}, \mathrm{Y}^t) \ + \ \alpha \|\mathbf{W}\|_F^2 + \beta \|\mathrm{S}\|_F^2, \tag{19}$$

where the first term is a loss function which is a linear combination of task-specific losses $\mathcal{L}^t$ with task weightings λ^t; the second term is to penalize the complexity of $\mathbf{W}$ which denotes all the trainable parameters of our model, and the third term is to penalize the complexity of the learned affinity matrix S. In our experiments, we use Vizier [8] to search for the best weighting schemes on the performance of MRAN as well as other DL-MTL approaches.

For each task, considering least-squares loss function, we optimize the following objective function:

$$\mathcal{L}^t(\mathrm{X}, \mathrm{Y}^t) = \sum_{i=1}^{n} \|\hat{y}_i^t - y_i^t\|_2^2, \tag{20}$$

where $\hat{y}_i^t$ is the prediction value of task $\mathcal{T}^t$ and y_i^t is the ground truth of task $\mathcal{T}^t$. And Adam [11] is employed as the optimizer.

3.7 Theoretical Analysis

In this section, we compare the proposed MRAN with two state-of-art methods, cross-stitch [17] and MMoE [16], which have been commonly used for MTL problems. Cross-Stitch model first trains each task separately, then combines the separately trained networks together by a cross-stitch unit. This unit learns a combination of task-specific representations for each task based on Equation (6). However, relationship matrix $\boldsymbol{\alpha}$ needs pre-definition, and initialization of $\boldsymbol{\alpha}$ has a significant impact on the results.

In contrast to cross-stitch, MRAN first learns a hidden representation of the studied tasks by the self-attention module and the hidden tasks can be represented as $\mathbf{H}_\mathrm{O} = \boldsymbol{\alpha}_\mathrm{O} \odot \mathrm{Y}$; then, MRAN learns task-specific embedding of features by aligning features to $\mathbf{H}$. If we replace the task relationship mapping $\boldsymbol{\alpha}_\mathrm{O}$ with affinity matrix as defined in Equation (6), and limit the number of tasks to two, MRAN reduces to cross-stitch method. From the comparison, it can be seen that cross-stitch is a particular case of MRAN.

Generally, DL-MTL models have adopted Shared-Bottom multi-task DNN structure. Instead, MMoE has a group of bottom networks, each of which is called an expert, and a gating network which assembles the experts with different weights to learn the relationships between input and output. In fact, the Mixture-of-Experts mechanism is equivalent to multi-head attention mechanism in MRAN$_m$ which learns different representation subspaces at different feature sets, and the gating mechanism is equivalent to location-based attention mechanism. However, the gating network in MMoE does not consider feature interactions.

The aforementioned theoretical analysis guarantees that our approach has great potential to obtain better performance.

4 EXPERIMENTS

In this section, we conduct experiments on two public datasets and one real-world dataset to validate the effectiveness of our proposed approaches. And then we analyze the impact of each key component in our proposed approaches on model performance. Codes and datasets will be released.

4.1 Dataset

We evaluate the performance of our proposed approaches on two public datasets: Climate dataset, Sarcos dataset, and one real-world dataset: Dianping dataset. For each dataset, we split 50% as training set and the rest 50% for testing.

- **Climate dataset**: This is a regression dataset[1] [19], which was collected by a sensor network consisting of four climate stations in the south of England: Cambermet, Chimet, Sotonmet, and Bramblemet. In this experiment, the normalized climate signal from 1 March, 2017 to 31 March, 2017, in 5-minute intervals (5,369 samples) are used as features, and predictions of air temperatures of the four stations are regarded as four learning tasks.

[1] The data can be obtained from www.cambermet.co.uk. and the sites therein.

- **Sarcos dataset**: This is a regression dataset [5] where the goal is to predict the torque measured at each joint of a 7 degrees-of-freedom robotic arm, given the current state, velocity, and acceleration measured at each joint (7 torques for 21-dimensional input). Following the procedure of [5], we have 7 tasks and 44,484 samples.
- **Dianping dataset**: This dataset describes the functions, rating and scores information of stores and malls in Beijing collected from a famous Chinese online review website DianPing.com[2]. Totally 456 shopping malls and 10,070 food chain stores inside are included. The goal is to predict the business success of food chain stores when placed in a candidate mall. The business success is reflected by the review number, the ratings, and scores of quality, environment, and service. Prediction of each business success criteria for stores is regarded as a task.

4.2 Evaluation Metrics

The evaluation measurements approach we used is the root mean squared error (*RMSE*) and Pearson correlation coefficient (*PCC*). The metrics for evaluating the performance of approaches include Root Mean Square Error (RMSE) and Pearson Correlation Coefficient (PCC). The RMSE measures the consistency of prediction results and real values with respect to L2-norm. The PCC measures the linear correlation of the predicted vector with target vector.

4.3 Baseline Methods

We compare our approaches with the following methods.

Traditional single task learning methods:

- **LASSO**. This single-task learning method learns each task independently with L1-norm regularization.
- **RDIDGE**. Learning each task independently with L2-norm regularization.

Regularization-based multi-task learning methods:

- **MTFL**. Multi-task feature learning by Argyriou et al. [1] enforces to share features across all tasks. It does not make assumptions on feature interactions.
- **MTIL**. Multi-task feature interaction learning [13] regularized by the $\ell_{2,1}$ norm of the weight matrix $\mathbf{W}$ and the tensor group lasso norm of interaction tensor $\mathcal{Q}$.

Deep learning based single task learning method:

- **NN**. A simple feed-forward neural network with a single hidden layer.

Deep learning based multi-task learning methods:

- **Cross-Stitch**. This method shares knowledge between two tasks by introducing a "Cross-Stitch" unit [17]. The cross-stitch unit has α values controlling the flow between layers of two neural networks.
- **MMoE**. This method adopts the multi-gate mixture-of-experts (MMoE) structure to multi-task learning [16]. The MMoE model has a group of bottom networks (each of which is called an expert) and multi-gate to allow different tasks to utilize experts differently.

4.4 Our Approaches

Our proposed MRAN model is a general MTL framework for predicting multiple tasks. We show the performance of the following four approaches in the experiments.

- MRAN$_d$. This approach is based on dot-product attention mechanism.
- MRAN$_g$. It is based on general attention model.
- MRAN$_c$. MRAN$_c$ uses concatenation attention mechanism.
- MRAN$_m$. Similar to the aforementioned approaches, MRAN$_m$ employs multi-head attention mechanism in prediction model.

4.5 Hyper-Parameter Tuning

We employ a Gaussian Process Bandits algorithm in Vizier [8] for tuning hyper-parameters, which is used in recent deep learning frameworks to search for the best hyper-parameters. We tune the hidden units per layer, learning rates and the number of training steps for all methods. To make the comparison fair, we set the same number of hidden units per layer for all deep learning methods. All deep learning methods are implemented using Pytorch. We also tune some method-specific hyper-parameters, e.g., number of heads in MRAN$_m$, number of experts, number of hidden units per expert in MMOE, and cross-stitch unit in Cross-Stitch.

4.6 Results of Model Performance

We show the performance of our proposed four approaches and baselines on Climate, Sarcos and Dianping datasets in terms of *RMSE* in Table 1, 2 and 3 respectively.

Table 1: Performance comparison among various approaches on the Climate dataset in terms of *RMSE.*

	y^1	y^2	y^3	y^4
LASSO	1.391	1.510	1.335	1.804
RIDGE	1.374	1.484	1.310	1.762
MTFL	1.380	1.482	1.311	1.775
MTIL	1.360	1.475	1.299	1.786
NN	1.405	1.451	1.268	1.585
Cross-Stitch	1.047	1.077	0.862	1.243
MMoE	0.946	1.055	1.041	1.602
MRAN$_d$	1.046	1.021	1.057	1.320
MRAN$_g$	0.994	1.024	0.865	1.313
MRAN$_c$	0.890	0.955	1.006	**1.224**
MRAN$_m$	**0.817**	**0.871**	**0.832**	1.236

First, from all of the aforementioned tables, we can see that MRAN$_m$ outperforms other models in all tasks, and the performance of MRAN$_c$, MRAN$_g$ and MRAN$_d$ outperform other models in most tasks. The average relative improvement of all tasks in MRAN$_m$ is encouraging with gains up to 59.12% (MTIL), 14.01% (Cross-Stitch) and 22.94% (MMoE) in Climate dataset, gains up to 6.35% (MTIL), 7.19% (Cross-Stitch) and 8.46% (MMoE) in Dianping dataset and gains up to 179.64% (MTIL), 85.23% (Cross-Stitch) and 45.76% (MMoE) in Sarcos dataset. Thus, MRAN$_m$ achieves the best results among all the proposed approaches and baselines.

Secondly, we found that all the MTL methods outperform the single task learning approaches (LASSO, RIDGE, and NN), which demonstrates the effectiveness of learning multiple tasks jointly by

[2] http://www.dianping.com/

Table 2: Performance comparison among various approaches on the Sarcos dataset in terms of *RMSE*.

	y^1	y^2	y^3	y^4	y^5	y^6	y^7
LASSO	5.738	5.021	3.136	3.340	0.645	1.081	0.820
RIDGE	5.571	4.813	3.014	3.113	0.373	0.902	0.665
MTFL	5.583	4.805	3.007	3.122	0.375	0.902	0.666
MTIL	5.537	4.744	2.981	3.117	0.355	0.893	0.659
NN	5.534	4.936	3.002	3.375	0.393	0.919	0.704
Cross-Stitch	3.774	3.456	1.749	1.885	0.315	0.553	0.403
MMoE	3.094	2.329	1.328	1.335	0.284	0.431	0.358
MRAN$_d$	2.959	2.185	1.106	1.163	**0.153**	0.298	0.249
MRAN$_g$	3.014	2.230	1.186	1.260	0.154	0.298	0.261
MRAN$_c$	2.955	2.255	1.188	1.121	0.177	0.303	0.285
MRAN$_m$	**2.879**	**1.895**	**1.041**	**0.829**	0.154	**0.261**	**0.236**

exploring the relationships between tasks. Additionally, the MMoE obtains better performance in Sarcos dataset which indicates that learning the better representations of all tasks benefit achieving better performance. Finally, the performance of MTIL is better than MTFL in most tasks, which indicates that considering feature interactions would be better.

Furthermore, Figures 5 (a), 5 (b) and 5 (c) show the *PCC* of our proposed approaches and baselines on Climate, Sarcos and Dianping datasets respectively. We can observe that our proposed four approaches (four dotted red lines at the top) outperform all other baselines in terms of *PCC* in most tasks. Note the MRAN$_m$ gives the best *PCC* in most of the tasks compared with other proposed approaches and baselines. This verifies that MRAN$_m$ is effective in learning the complicated T-T transfer relationship and F-F interaction relationship by multi-head attention mechanism. These experimental results are consistent with the theoretical analysis.

Table 3: Performance comparison among various approaches on the Dianping dataset in terms of *RMSE*.

	y^1	y^2	y^3	y^4	y^5
LASSO	1182.715	3.917	0.596	0.625	0.648
RIDGE	1202.650	3.724	0.491	0.494	0.536
MTFL	1163.497	3.712	0.490	0.495	0.535
MTIL	1175.880	3.689	0.486	0.494	0.526
NN	1150.603	4.510	0.557	0.594	0.590
Cross-Stitch	1087.358	3.700	0.504	0.493	0.572
MMoE	1083.034	3.971	0.500	0.513	0.552
MRAN$_d$	1034.700	3.628	0.485	0.543	0.526
MRAN$_g$	1070.898	3.671	0.483	0.482	0.515
MRAN$_c$	1030.112	3.611	0.481	**0.475**	**0.514**
MRAN$_m$	**988.206**	**3.559**	**0.470**	0.477	0.515

4.7 Effect of Components

To understand the performance of MRAN, we analyze the effect of each key component of four proposed approaches on Dianping dataset. As illustrated in table 4, the first group rows are the results of MRAN with only F-F interaction relationship learning module that we consider as the basic model (C1). Comparing the complete MRAN model, take MRAN$_m$ as an example, the *RMSE* enhances by 63.014, 0.049, 0.012, 0.012, and 0.014 in each task respectively.

By adding the T-F dependence relationship learning module following the F-F interaction relationship learning module, we build the C2 model. Compared with the C1 model, the *RMSE* of the C2 model drops by 48.626 in task 1 and 0.058 in task 2. That is a significant improvement, and it is because the better task-specific representations are learned through the C2 model. By additionally employing the self-attention mechanism to learn T-T transfer relationship, we build the complete model of MRAN, also called the C3 model. Compared with the C2 model, the C3 model achieves further improvement in *RMSE*. Figure 6 shows the performance of MRAN in terms of *PCC*. As we can see, with the number of modules increasing, the *PCC* increases. According to the results, all of the components positively contribute to the final performance.

Table 4: Effect of components on our approaches in terms of *RMSE*.

Model	Attention type	y^1	y^2	y^3	y^4	y^5
Only F-F interaction (C1)	Dot	1052.824	3.819	0.523	0.526	0.558
	General	1093.707	3.748	0.518	0.508	0.550
	Concat	1039.900	3.679	0.512	0.507	0.546
	Multi-head	1051.221	3.608	0.482	0.489	0.529
+T-F dependence (C2)	Dot	1049.312	3.732	0.489	0.508	0.533
	General	1092.656	3.648	0.485	0.484	0.519
	Concat	1035.403	3.633	0.481	0.486	0.517
	Multi-head	1036.833	3.617	0.475	0.477	0.516
+ T-T transfer relationship (C3)	Dot	1034.700	3.628	0.485	0.543	0.526
	General	1070.898	3.671	0.483	0.482	0.515
	Concat	1030.112	3.611	0.481	**0.475**	**0.514**
	Multi-head	**988.206**	**3.559**	**0.470**	0.477	0.515

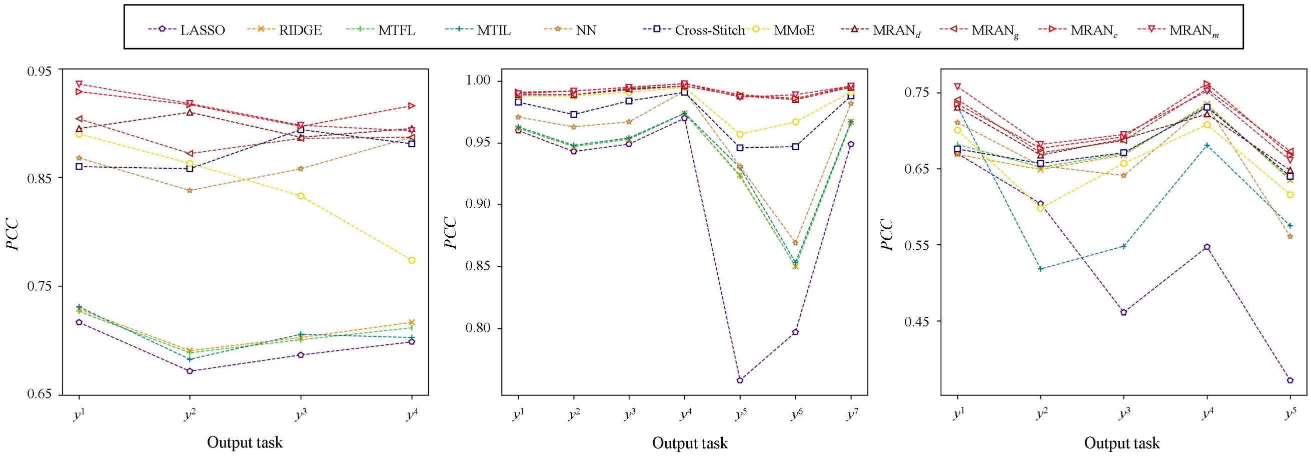
5 RELATED WORK

5.1 Deep Learning based Multi-task Learning

DL-MTL provides a convenient way of combining information from multiple tasks due to its capacity so that it can result in improved efficiency for each task [3, 7, 17, 29]. Typical DL-MTL models were proposed by Caruana [3], which shared the bottom layers for all tasks and split top layers for each task.

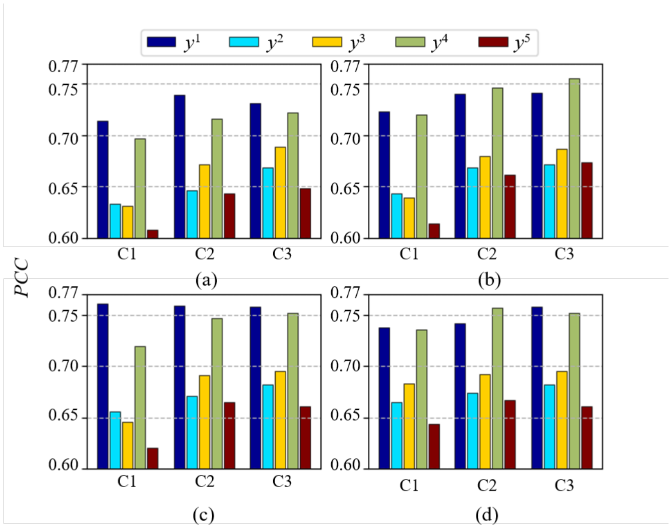
Recently, there have been some efforts on finding more meaningful partly sharing structures between tasks instead of by simply sharing hidden layers fully. Duong et al. [7] used regularization constraints parameters sharing between the source and target language. The cross-stitch network [17] learned an optimal combination of task-specific representations for each task. Lee et al. [12] considered asymmetry in feature transfer between the related tasks. Yang et al. [26] used a tensor factorization model to generate hidden-layer parameters for each task. Ma et al. [16] added multi-gate for each task to capture the task differences and model the task relationships. Cao et al. [2] designed a partially shared multi-task convolutional neural network to learn better shared and task-specific representations.

Compared to typical DL-MTL methods, these approaches learned better shared and task-specific features and can achieve better performance. However, the proposed approaches in [2, 7] may rely on a specific application scenario and the proposed approach in [17] can deal with few tasks. More importantly, these approaches



(a) (b) (c)

Figure 5: Performance comparison among various approaches on the Climate dataset (a), Sarcos dataset (b) and Dianping dataset (c) in terms of *PCC*.



(a) (b) (c) (d)

Figure 6: Effect of components on our approaches in terms of *PCC*. (a) MRAN$_d$ method; (b) MRAN$_g$ method; (c) MRAN$_c$ method; (d) MRAN$_m$ method.

enabled knowledge to transfer in the original feature space but did not consider in feature interaction space.

5.2 Feature Interaction

Traditional machine learning methods have focused on learning a linear prediction model which can be called *linear effects*. Apart from the linear effects, an important but usually ignored issue is feature interaction. Recently, many advance studies in supervised learning have suggested considering feature interaction into regression and classification models which could help capture the non-linearity of the data, thus enhancing the performance [13, 28].

In [4], the authors extended the LASSO method for simultaneously fitting a regression model and identifying the important interaction effects. Zeng et al. [28] proposed the IWFS method to measure the redundancy and interaction of features through the formalized concept of feature relevance, redundancy, and interaction. However, the IWFS is limited to account for interactions of three features at most and the proposed method in [4] may not scale to high dimensional feature space. To address the problem, Shishkin et al. [22] developed the CMICOT method which was based on conditional mutual information (MI), and used a greedy approximation and binary representatives to mitigate this cost.

In many real scenarios, there are multiple related tasks which involve interaction effects. Recent [13] considered feature interaction representations and task relatedness over feature interactions in MTL based on regularization framework with a single output. Sun et al. [24] proposed Subspace Network which aimed to model non-linear interactions among the variables for multi-task censored regression problem. In our work, we further addressed this problem with a new perspective by modeling a better feature representation for all tasks, which considers the different interaction effects among different feature subspaces.

5.3 Attention-based Neural Networks

The idea of neural attention mechanism is loosely based on human visual attention, which has been successfully used in various applications [15, 27]. Specifically, attention mechanism has been used to select the focus of source sentence for improving neural machine translation [15]. It provides a natural way to identify related features for different tasks in MTL. However, few studies have been done along this route, which is one of the focus of this work.

Self-attention is a particular case of attention mechanism. It can relate different positions of a single sequence to computing a representation of the sequence. The advantages of self-attention are: 1) conducting direct connections between two arbitrary tokens in a sentence; 2) providing a more flexible way to select, represent and synthesize the information of sentences. Self-attention has been used successfully in many applications including natural language inference, and learning task-independent sentence representation [6, 14, 25]. To the best of our knowledge, this is the first attempt to model task relationships and feature-feature interactions by self-attention mechanism, which leads to an intrinsic representation of tasks and features, and therefore has great potential to improve the performance of MTL.

6 CONCLUSION

This paper provides a comprehensive deep learning based MTL framework, MRAN, which is able to improve the prediction accuracy by leveraging multiple relationships. Compared with previous MTL methods, MRAN can not only control positive and negative knowledge transfer among tasks adaptively but also benefits from multiple relationship learning. In particular, to differentiate the knowledge transfer among tasks, a self-attention based task relationship learning module is provided firstly. Then, to capture the interactions of features, a feature-feature relationship learning module is designed. Last, to specify different features for different tasks, an attention-based task-feature alignment mechanism is developed. For the purpose of evaluating the proposed MRAN, we conducted experiments on two public datasets and one real-world dataset. Both classical and the state-of-the-art MTL methods are employed to provide benchmark performance. Experimental results demonstrated the superiority of MRAN over these baselines. This paper enriches the MTL research from two perspectives: 1) an adaptive task relationship learning mechanism, which avoids negative knowledge transfer among tasks caused by improperly predefined task relationships; 2) a comprehensive DL-MTL framework, which improves the prediction performance by leveraging multiple self-taught relationships among features and tasks.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments on this research work. This work is supported by the National Natural Science Foundation of China under Grant No. 51822802, 51778033, U1811463, 91746301, U1836206, and 61773361, the Science and Technology Major Project of Beijing under Grant No. Z171100005117001 and the National Key R & D Program of China under Grant No. 2016YFC0801700.

REFERENCES

[1] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2008. Convex multi-task feature learning. *Machine Learning* 73, 3 (2008), 243–272.

[2] Jiajiong Cao, Yingming Li, and Zhongfei Zhang. 2018. Partially Shared Multi-Task Convolutional Neural Network with Local Constraint for Face Attribute Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 4290–4299.

[3] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.

[4] Nam Hee Choi, William Li, and Ji Zhu. 2010. Variable selection with the strong heredity constraint and its oracle property. *J. Amer. Statist. Assoc.* 105, 489 (2010), 354–364.

[5] Carlo Ciliberto, Alessandro Rudi, Lorenzo Rosasco, and Massimiliano Pontil. 2017. Consistent multitask learning with nonlinear output relations. In *Advances in Neural Information Processing Systems.* 1986–1996.

[6] Chaoqun Duan, Lei Cui, Xinchi Chen, Furu Wei, Conghui Zhu, and Tiejun Zhao. 2018. Attention-Fused Deep Matching Network for Natural Language Inference. In *IJCAI.* 4033–4040.

[7] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low Resource Dependency Parsing: Cross-lingual Parameter Sharing in a Neural Network Parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, Vol. 2. 845–850.

[8] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. 2017. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 1487–1495.

[9] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[10] Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. 2017. One model to learn them all. *arXiv preprint arXiv:1706.05137* (2017).

[11] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[12] Haebeom Lee, Eunho Yang, and Sung Ju Hwang. 2017. Deep Asymmetric Multi-task Feature Learning. *arXiv preprint arXiv:1708.00260* (2017).

[13] Kaixiang Lin, Jianpeng Xu, Inci M Baytas, Shuiwang Ji, and Jiayu Zhou. 2016. Multi-task feature interaction learning. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 1735–1744.

[14] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130* (2017).

[15] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).

[16] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* ACM, 1930–1939.

[17] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 3994–4003.

[18] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10).* 807–814.

[19] Gabriel Parra and Felipe Tobar. 2017. Spectral mixture kernels for multi-output Gaussian processes. In *Advances in Neural Information Processing Systems.* 6681–6690.

[20] Jiaming Shen, Maryam Karimzadehgan, Michael Bendersky, Zhen Qin, and Donald Metzler. 2018. Multi-Task Learning for Email Search Ranking with Auxiliary Query Clustering. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management.* ACM, 2127–2135.

[21] Timothy J Shields, Mohamed R Amer, Max Ehrlich, and Amir Tamrakar. 2017. Action-Affect-Gender Classification Using Multi-task Representation Learning.. In *CVPR Workshops.* 2249–2258.

[22] Alexander Shishkin, Anastasia Bezzubtseva, Alexey Drutsa, Ilia Shishkov, Ekaterina Gladkikh, Gleb Gusev, and Pavel Serdyukov. 2016. Efficient high-order interaction-aware feature selection based on conditional mutual information. In *Advances in Neural Information Processing Systems.* 4637–4645.

[23] Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vol. 2. 231–235.

[24] Mengying Sun, Inci M Baytas, Liang Zhan, Zhangyang Wang, and Jiayu Zhou. 2018. Subspace Network: Deep Multi-Task Censored Regression for Modeling Neurodegenerative Diseases. *arXiv preprint arXiv:1802.06516* (2018).

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems.* 5998–6008.

[26] Yongxin Yang and Timothy Hospedales. 2016. Deep multi-task representation learning: A tensor factorisation approach. *arXiv preprint arXiv:1605.06391* (2016).

[27] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* 1480–1489.

[28] Zilin Zeng, Hongjun Zhang, Rui Zhang, and Chengxiang Yin. 2015. A novel feature selection method considering feature interaction. *Pattern Recognition* 48, 8 (2015), 2656–2666.

[29] Liang Zhang, Keli Xiao, Hengshu Zhu, Chuanren Liu, Jingyuan Yang, and Bo Jin. 2018. CADEN: A Context-Aware Deep Embedding Network for Financial Opinions Mining. In *2018 IEEE International Conference on Data Mining (ICDM).* IEEE, 757–766.

[30] Yu Zhang and Qiang Yang. 2017. A Survey on Multi-Task Learning. *CoRR* abs/1707.08114 (2017). http://arxiv.org/abs/1707.08114

[31] Yu Zhang and Dit-Yan Yeung. 2012. A convex formulation for learning task relationships in multi-task learning. *arXiv preprint arXiv:1203.3536* (2012).