Saliency-Regularized Deep Multi-Task Learning

Guangji Bai
Emory University
Atlanta, GA
guangji.bai@emory.edu

Liang Zhao
Emory University
Atlanta, GA
liang.zhao@emory.edu

ABSTRACT

Multi-task learning (MTL) is a framework that enforces multiple learning tasks to share their knowledge to improve their generalization abilities. While shallow multi-task learning can learn task relations, it can only handle pre-defined features. Modern deep multi-task learning can jointly learn latent features and task sharing, but they are obscure in task relation. Also, they pre-define which layers and neurons should share across tasks and cannot learn adaptively. To address these challenges, this paper proposes a new multi-task learning framework that jointly learns latent features and explicit task relations by complementing the strength of existing shallow and deep multitask learning scenarios. Specifically, we propose to model the task relation as the similarity between tasks' input gradients, with a theoretical analysis of their equivalency. In addition, we innovatively propose a multi-task learning objective that explicitly learns task relations by a new regularizer. Theoretical analysis shows that the generalizability error has been reduced thanks to the proposed regularizer. Extensive experiments on several multi-task learning and image classification benchmarks demonstrate the proposed method's effectiveness, efficiency as well as reasonableness in the learned task relation patterns.

KEYWORDS

Multi-task Learning, Task Relation, Saliency Detection

ACM Reference Format:
Guangji Bai and Liang Zhao. 2019. Saliency-Regularized Deep Multi-Task Learning. In *Proceedings of ACM Conference (KDD'22).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Multi-task learning (MTL, [9]) is an important research domain based on the idea that the performance of one task can be improved using related tasks as inductive bias. While traditional shallow MTL methods can fit the models for individual tasks and learn task relations, they do not focus on generating features from scratch and instead rely on pre-defined and explicit features [51]. More recently, deep representation learning empowers MTL to go "deep" by equipping it with the capacity to generate features while fitting the tasks' predictive models. Deep MTL is usually categorized according to the ways of correlating tasks' models into two major types: *hard-parameter sharing* and *soft-parameter sharing.* Hard-parameter sharing methods [28, 39, 53] essentially hard-code which part of neurons or layers to share and which part does not for different tasks instead of doing it adaptively. They usually share the layers for representation learning (e.g., convolutional layers) but not those for decision making (e.g., fully-connected layers for classification). On the other hand, soft-parameter sharing methods [12, 33, 48] do not require to hard-code the sharing pattern but instead build individual models for each task and "softly" regularize the relatedness among them. Hence, soft-parameter sharing has better flexibility in learning the task relation, while may not be efficient since its model parameters increase linearly with the number of tasks. Hard-parameter sharing, by contrast, is more "concise" but requires pre-define which parts are shared or not.

Therefore, although MTL is a long-lasting research domain, it remains a highly challenging and open domain that requires significantly more effort to address challenges such as the trade-off between model flexibility and conciseness of hard- and soft-parameter sharing mentioned above. Although more recently, there have come a few attempts trying to alleviate the dilemma, such as those regularizing task relationships in task-specific layers in hard-parameter sharing to achieve knowledge transfer in unshared layers [28] and those adaptively learning which part to share or not by methods like branching [18, 29] or Neural Architecture Search [44], the research frontiers still suffer from several critical bottlenecks, including **(1) Difficulty in regularizing deep non-linear functions of different tasks.** Adaptively learning task relation requires regularizing different tasks' predictive functions, which, however, are much harder to achieve for nonlinear-nonparametric functions since it requires regularizing in the whole continuous domain of input. To work around it, existing works [28, 43] typically resort to a *reduced problem* which is to regularize the neural network parameters. Notice this reduction deviates from the original problem and is over-restricted. For example, first, two neural networks with different permutations of latent neurons can represent the same function [8]. Moreover, even if they have different architectures, they can still possibly represent the same function [23]. This gap deteriorates the model's generalizability and effectiveness. **(2) Lack of interpretability in joint feature generation and task relation learning.** Despite incapability of generating features, shallow MTL enjoys good interpretability since they learn explicit task correlations via the way how the hand-crafted features are utilized in the tasks. However, in deep MTL, the generated features do not have explicit meaning and how the black-box models relate to each other is highly obscure. It is imperative yet challenging to increase the interpretability of both generated features and task relation. **(3) Difficulty in theoretical analysis.** While there are fruitful theoretical analyses on shallow MTL, such as on generalization error [4–6] and conditions for regularized MTL algorithms to satisfy representer theorems [2, 3], similar analyses meet strong hurdles to be extended to deep MTL due to the difficulty in reasoning about neural networks whose feature space is given by layer-wise embeddings [47]. It is important to enhance the theoretical analyses on

KDD'22, Aug 14-18 2022, Washington DC, U.S.
2019. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

the model capacity and theoretical relation among different deep MTL models.

This paper proposes a new Saliency-regularized Deep Multi-task Learning (SRDML) framework to solve the challenges mentioned above. First, we reconsider the feature weights in traditional linear multitask learning as the input gradient and then generalize the feature learning into the non-linear situation by borrowing the notion of saliency detection. Second, we recast the task relation problem as the similarity among saliency regions across tasks so as to regularize and infer the task relation. Third, to validate our hypothesis, we have given a theoretical analysis of their equivalency. Meanwhile, we also provide theoretical analysis on how the proposed regularization helps reduce the generalization error. Finally, we demonstrate our model's effectiveness and efficiency on synthetic and multiple large-scale real-world datasets under comparison with various baselines.

2 RELATED WORK

Multi-task learning (MTL). Readers may refer to [11, 51] for a more comprehensive survey on MTL. Before the popularity of deep learning, traditional MTL usually focuses on hand-crafted features and can be generally divided into two categories: **1).** *multi-task feature learning*, which aims to learn a shared/similar feature selection, latent space, or model parameters [1, 14]. **2).** *multi-task relation learning*, which aims to quantify task relatedness via task clustering [19, 22] or task co-variance [16, 52]. However, they rely on hand-crafted features and the separation of feature generation and task learning may result in sub-optimal performance.

More recently, MTL takes advantage of the advancement of deep neural networks which can directly take raw, complex data such as images, audio, and texts and learn in an end-to-end manner. Deep MTL integrates feature generation and task learning and simultaneously learns both of them [7]. In this domain, *hard-parameter sharing* [34, 53] requires to hard-code which part of the network is shared and which is not. Existing work usually shares the lower-level layers for representation learning (e.g., convolutions) while make higher-level layers (e.g., those for classification) separated across tasks. Some existing works extend hard-parameter sharing by considering Neural Architecture Search [13] like [29], [44], and [18]. *soft-parameter sharing* based method has better flexibility where each task has its own models and regularization is used to enforce task relatedness by aligning their model parameters [12, 33, 48]. To achieve both hard-parameter sharing's conciseness and efficiency and soft-parameter sharing's flexibility, some recent work [28, 43] shares the representation learning layers while exploits task relations in task-specific layers.

Saliency detection. Saliency detection is to identify the most important and informative part of input features. It has been applied to various domains including CV [17, 37], NLP [24, 35], etc. The salience map approach is exemplified by [50] to test a network with portions of the input occluded to create a map showing which parts of the data actually have an influence on the network output. In [40], a salience map can be created by directly computing the input gradient. Since such derivatives can miss important aspects of the information that flows through a network, a number of other approaches have been designed to propagate quantities other than gradients through the network. For example, in the vision domain, Class Activation Mapping (CAM, [55]) modifies image classification CNN architectures by replacing fully-connected layers with convolutional layers and global average pooling [25], thus achieving class-specific feature maps. Grad-CAM [38] generalizes CAM by visualizing the linear combination of the last convolutional layer's feature map activations and label-specific weights, which are calculated by the gradient of prediction score w.r.t the feature map activations. Grad-CAM invokes different versions of backpropagation and/or activation, which results in aesthetically pleasing, heuristic explanations of image saliency. While there exist some other saliency-based methods along this research line including Guided Propagation [42], Deconvolutional Network [50], etc, they are designed only for specific architectures like ReLU Network for Guided Propagation.

	Transparency in task relation	Feature generation	Conciseness and efficiency	Flexibility
Shallow MTL	✓	✗	✓	✗
Hard-parameter sharing	✗	✓	✓	✗
Soft-parameter sharing	✗	✓	✗	✓
SRDML	✓	✓	✓	✓

Figure 1: Comparison over different MTL methods.

3 PROPOSED METHOD

In this section, we introduce our proposed Saliency-regularized Deep Multi-task Learning (SRDML) method. We first review the pros and cons for each MTL method and describe our main motivation, then formally introduce our model and its objective function.

3.1 Problem Formulation

Consider a multi-task learning problem with T tasks such that a dataset $\{\mathbf{X}, \mathbf{Y}_1, \mathbf{Y}_2, \cdots, \mathbf{Y}_T\}$ is given with i.i.d training samples $\mathbf{X}_t = \{\mathbf{x}_1^{(t)}, \mathbf{x}_2^{(t)}, \cdots, \mathbf{x}_n^{(t)}\}$, $\mathbf{Y}_t = \{\mathbf{y}_1^{(t)}, \mathbf{y}_2^{(t)}, \cdots, \mathbf{y}_n^{(t)}\}$, where n is the sample size and $(\mathbf{x}_i^{(t)}, \mathbf{y}_i^{(t)})$ is a pair of input and label such that $\mathbf{x}_i^{(t)} \in \mathcal{X}$ and $\mathbf{y}_i^{(t)} \in \mathbb{R}$, $\forall\ i = 1, 2, \cdots, n$ and $t = 1, 2, \cdots, T$.

Mathematically, consider a predictor g which factorizes as $g = f \circ h$, where "$\circ$" stands for functional composition. The function $h : \mathcal{X} \to \mathbb{R}^K$ is called the feature or representation extraction part and is shared for all tasks, while $f : \mathbb{R}^K \to \mathbb{R}$ is a function defined on $\mathbb{R}^K$, a predictor specialized to each task at hand. K here denotes the latent representation or feature-map dimensions. Following existing work like [28, 39], we assume each task shares the same input feature $\mathbf{x}$, i.e., $\mathbf{x}^{(1)} = \mathbf{x}^{(1)} = \cdots = \mathbf{x}^{(T)}$, which is very commonly seen in deep MTL problems such as multi-task image classification task in computer vision domain.

Our goal is to build a deep architecture for learning multiple tasks $\mathbf{y}_i^{(t)} = g_t(\mathbf{x}_i)$, $t \in \{1, \ldots, T\}$ which jointly generates semantic features and learns task relation to correlate different tasks with interpretability. This goal poses significant challenges to existing work: **1).** Directly regularizing the prediction function of different tasks is extremely hard. Existing work considered a reduced

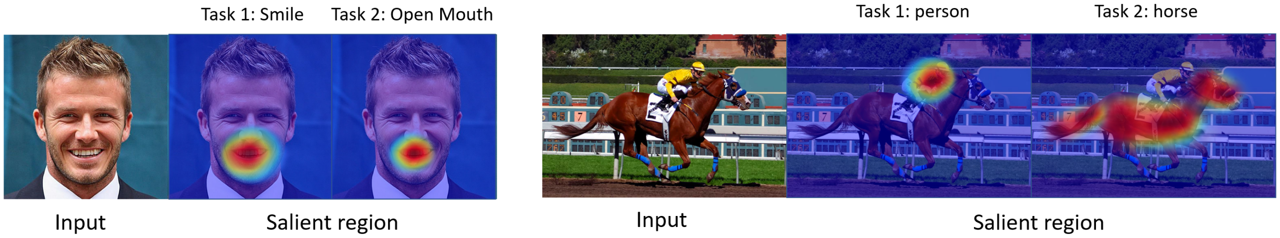


Figure 2: Relation between saliency and task similarity. Left: Two tasks are to detect whether the man is smiling and his mouth is open. The salient regions for two tasks are both around the mouth. Right: Two tasks are to detect the horse and person. The salient regions are close to each other, indicating the potential similarity between the tasks.

problem by regularizing the feature weights of different f_t which is over-restricted. **2).** How to learn interpretable task relations with deep/implicit features is still unclear. **3).** Theoretical analysis is rare in deep MTL due to the non-linear and non-parametric functions of h and f. To solve these challenges, we reconsider the feature weights in shallow MTL as input gradient, i.e., $\partial f(x)/\partial x,\ x \in \mathbb{R}^K$, and generalize the feature learning into the deep network by saliency detection method.

3.2 Motivations

We propose a simple framework that can innovatively achieve all the goals, as shown in Figure 1.

To achieve model conciseness and efficiency as well as task relatedness flexibility, we share the representation learning layers and learn task relationships in task-specific layers. This is based on important neuro-inspirations: human sensory organs and retina are the same for all different tasks (meaning the convolution layers are shared). On the other hand, the working memory will leverage the long-term memory for each task and related tasks will have related memory (i.e., model) and their relatedness can be considered as the similarities of activation patterns for different tasks, namely the similarity among the saliency maps for different tasks.

Then, the next question is how to regularize the relation among different tasks, namely how to regularize the (dis)similarity of the predictive functions of different tasks. As mentioned above, it is problematic to directly regularize the neuron network parameters due to their gap with the actual function. For example, neural networks with different architecture or neuron permutation could represent the same function. Therefore, this motivates us to explore an innovative alternative so that we can more easily work towards the space of *functional*. Specifically, we propose to regularize first-order derivatives with respect to the input of different tasks. This new strategy has two crucial merits: First, it is mathematically equivalent to directly regularize the function without the gap in existing works mentioned above. Second, it also finds inspiration from the saliency map domain and comes with strong interpretability in how tasks correlate.

Key Merit 1: Regularizing task functions without theoretical gap. Specifically, Theorem 1 below tells us that enforcing multiple tasks to have similar input gradients is equivalent to encouraging those tasks themselves to be similar.

Theorem 1. Define $\mathcal{F} := \{f \in \mathbf{C}^1 : f(0) = 0\}$, where $\mathbf{C}^k$ is the family of functions with k^{th}-order continuous derivatives for any non-negative integer k. Given $f_1, f_2 \in \mathcal{F}$, we have:

$$f_1 = f_2 \quad \textbf{if and only if} \quad f_1'(x) = f_2'(x),\ \forall x \in \mathcal{X} \tag{1}$$

Proof. Suppose $\mathcal{X} \subseteq \mathbb{R}^K$ is an open set and $f_1, f_2 : \mathcal{X} \to \mathbb{R}$, where both functions are differentiable and equal to zero at the origin. "$\Longrightarrow$": This direction is obvious, since two exactly the same functions will have the same gradient at any input point. "$\Longleftarrow$": Given $\nabla f_1(\mathbf{x}) = \nabla f_2(\mathbf{x})$, we know that $\partial f_1/\partial x_k = \partial f_2/\partial x_k,\ k = 1, 2, \cdots, K, \forall \mathbf{x} \in \mathcal{X}$. For arbitrary k, by $\partial f_1/\partial x_k = \partial f_2/\partial x_k$, we know that $\exists\ c_k(x_1, \cdots, x_{k-1}, x_{k+1}, \cdots, x_K),\ s.t.,\ f_1 = f_2 + c_k$. Meanwhile, notice $\forall\ l \neq k,\quad \partial c_k/\partial x_l = 0$ (otherwise, contradiction!) Hence, $dc_k = 0$ and we know c_k is a constant. Also, the value of c_k does not depend on k since for all k, l, we have $f_1 - f_2 = c_k = c_l$, thus there exists a constant c such that $f_1 = f_2 + c$. Finally, by the boundary condition that $f_1(\mathbf{0}) = f_2(\mathbf{0}) = 0$, we know that $c = 0$, i.e., $f_1 = f_2$, which finishes the proof. □

The assumption over function family $\mathcal{F} := \{f \in \mathbf{C}^1 : f(0) = 0\}$ is reasonable in practice since an all-zero input x simply corresponds to a black picture and for any tasks we assume a black picture contain no useful information and should be classified as the negative sample, i.e., with label 0.

Key Merit 2: Inspiration from saliency map and enhancement of interpretability. Evaluating task relation with derivative similarity has justification from saliency perspective. A saliency is a derivative of the prediction score w.r.t. input features, and it denotes the semantic features that influence the prediction most. In addition, similar tasks tend to have similar saliency, while dissimilar tasks tend to have dissimilar saliency. As shown in Figure 2, we enforce higher-level semantic features as saliency.

Many previous work have asserted that deeper representations in a CNN capture higher-level visual constructs [7, 31]. Furthermore, convolutional layers naturally retain spatial information which is lost in fully connected layers, so we expect the last convolutional layers to have the best compromise between high-level semantics and detailed spatial information. By following a recent work called Grad-CAM [38], we use the gradient information flowing into the last convolutional layer of the CNN to capture the saliency map to each neuron for a particular task or class of interests.

3.3 Objective function

We first give a formal definition of saliency. For example, in computer vision, given an input image I, a classification ConvNet f predicts I belongs to class c and produces the class score $f_c(I)$ (short

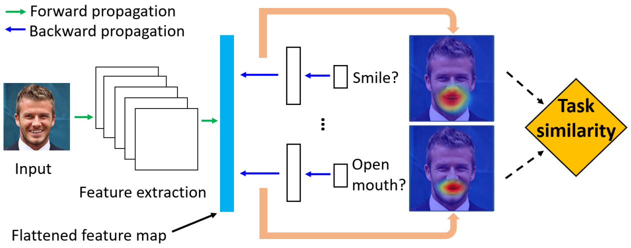


Figure 3: A high level overview of SRDML architecture.

as f_c). Let A be the feature map activations of the last convolutional layer. We are curious about the rank of each pixel in A based on their importance, which is referred to as saliency. The relationship between f_c and A is highly non-linear due to the non-linearity in f. In this case, we use the first-order derivatives i.e., $\partial f_c / \partial A$ to approximate the saliency map, which basically reflects the contributions of different pixels in A to the prediction f_c.

Now, we formally introduce the learning objective for SRDML as follows:

$$\begin{aligned} &\min_{h, f_1, \cdots, f_T, \xi} \textstyle\sum_{t=1}^{T} \mathcal{L}_t(f_t(h(\mathbf{X})), \mathbf{Y}_t), \text{ s.t.} \\ &\forall\ i, j,\ dist(\nabla_A f_i, \nabla_A f_j) \leq \xi_{ij}\ ,\ \textstyle\sum_{1 \leq i < j \leq T} \xi_{ij} \leq \alpha \end{aligned} \tag{2}$$

where i, j are task indexes with $1 \leq i < j \leq T$, $A = h(\mathbf{X})$ is the feature map activations from the last convolutional layer of h, and $\nabla_A f_t$ is the first-order derivative of function f_t with respect to A, i.e., $\partial f_t / \partial A$. The $dist()$ function here can be any distance measure including commonly-used ones like ℓ_1, ℓ_2, etc, and any potential normalization on the input gradient can also be embeded in $dist()$.

To adaptively learn the task relations, we introduce $\{\xi_{ij}\}_{1 \leq i < j \leq T}$, which is a set of learnable *slack* variables for each pair of tasks and α is a hyperparameter for controlling the overall level of slacking. Notice each ξ_{ij} can only take non-negative value and this is guaranteed by the inequality constraint and the non-negative norm.

Directly optimizing Eq. (2) could be difficult due to the constraint. By utilizing Lagrangian method, we further transform Eq. (2) into a regularized form as follows,

$$\begin{aligned} \min_{h, f_1, \cdots, f_T, \omega} &\ \textstyle\sum_{t=1}^{T} \mathcal{L}_t(f_t(h(\mathbf{X})), \mathbf{Y}_t) \\ &+ \lambda \cdot \textstyle\sum_{1 \leq i < j \leq T} \omega_{ij} \cdot dist(\nabla_A f_i, \nabla_A f_j) \\ &\text{s.t., } \omega_{ij} \geq 0\ \forall i, j,\ and\ \textstyle\sum_{1 \leq i < j \leq T} \omega_{ij} \geq \beta \end{aligned} \tag{3}$$

where $\{\omega_{ij}\}_{1 \leq i < j \leq T}$ is another set of learnable parameters to explicitly model task relationship during the multi-task training, and λ is the regularization coefficient. You may regard (3) follows the graph regularization [15, 16], where each node corresponds to a task and ω_{ij} represents the weight for the edge between task i and task j.

We further simplify the constraints by applying normalization onto $\{\omega_{ij}\}_{1 \leq i < j \leq T}$ as follow,

$$\begin{aligned} \min_{h, f_1, \cdots, f_T, \omega > 0} &\ \textstyle\sum_{t=1}^{T} \mathcal{L}_t(f_t(h(\mathbf{X})), \mathbf{Y}) \\ &+ \lambda \cdot \textstyle\sum_{1 \leq i < j \leq T} \dfrac{\omega_{ij}}{W} \cdot dist(\nabla_A f_i, \nabla_A f_j) \end{aligned} \tag{4}$$

where $W = \sum_{1 \leq i < j \leq T} \omega_{ij}$ is the sum of each ω_{ij}.

A general overview of our SRDML can be found in Figure 3.

4 THEORETICAL ANALYSIS

In this section, we present the theoretical analyses of our SRDML model. First, we prove our proposed regularizer can help reduce the generalization error. Second, we formally analyze the relation between SRDML and other MTL methods.

4.1 Generalization Error Bound

Here we show the generalization bound of our model. The major contribution is we theoretically proved our proposed regularization term can help reduce the generalization error.

Recall Eq. (2), where SRDML solves the following constrained problem:

$$\begin{aligned} &\min_{h, f_1, \cdots, f_T, \xi} \frac{1}{nT} \textstyle\sum_{t=1}^{T} \sum_{i=1}^{n} \mathcal{L}_t(f_t(h(\mathbf{x}_i)), \mathbf{y}_i^{(t)}), \text{ s.t.} \\ &\forall\ i, j,\ dist(\nabla_A f_i, \nabla_A f_j) \leq \xi_{ij}\ ,\ \textstyle\sum_{1 \leq i < j \leq T} \xi_{ij} \leq \alpha \end{aligned} \tag{5}$$

For simpler notation, define

$$\begin{aligned} \mathcal{F}_\epsilon :=\{ &\mathbf{f} \in \mathcal{F}^T\ :\ \forall\ 1 \leq i < j \leq T,\ x \in \mathcal{X}, \\ &dist(\nabla_x f_i, \nabla_x f_j) \leq \epsilon_{pq},\ \textstyle\sum_{1 \leq i < j \leq T} \epsilon_{ij} \leq \alpha \} \end{aligned} \tag{6}$$

where $\mathbf{f} = (f_1, f_2, \cdots, f_T)$ is the vectorization of each task's function, and $\{\epsilon_{ij}\}_{1 \leq i < j \leq T}$ is a set of global slack variables. Hence, the optimization problem can be simplified as

$$\min_{h \in \mathcal{H}, \mathbf{f} \in \mathcal{F}_\epsilon} \frac{1}{nT} \textstyle\sum_{t=1}^{T} \sum_{i=1}^{n} \mathcal{L}_t(f_t(h(\mathbf{x}_i)), \mathbf{y}_i^{(t)}) \tag{7}$$

Assumption 1. The loss function $\mathcal{L}$ has values in $[0, 1]$ and has Lipschitz constant 1 in the first argument, i.e.: (1) $\mathcal{L}(x, y) \in [0, 1]$ (2) $\mathcal{L}(x, y) \leq x, \forall y$.

Definition 1 (Expected risk, Empirical risk)**.** Given any set of function $h, f_1, \cdots, f_T$, we denote the expected risk as:

$$\mathcal{E}(h, f_1, \cdots, f_T) := \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{(X,Y) \sim \mu_t} [\mathcal{L}_t(f_t(h(X)), Y)] \tag{8}$$

Given the data $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$, the empirical risk is defined as:

$$\hat{\mathcal{E}}(h, f_1, \cdots, f_T | \mathbf{Z}) := \frac{1}{T} \sum_{t=1}^{T} \frac{1}{n} \textstyle\sum_{i=1}^{n} \mathcal{L}_t(f_t(h(\mathbf{x}_i)), \mathbf{y}_i^{(t)}) \tag{9}$$

Definition 2 (Global optimal solution, Optimized solution)**.** Denote $(h^*, \mathbf{f}^*)$ as the global optimal solution of the expected risk: $(h^*, \mathbf{f}^*) := \arg\min_{h \in \mathcal{H}, \mathbf{f} \in \mathcal{F}_\epsilon} \mathcal{E}(h, f_1, \cdots, f_T)$, and denote $(\hat{h}, \hat{\mathbf{f}})$ as the optimized solution by minimizing the empirical risk: $(\hat{h}, \hat{\mathbf{f}}) := \arg\min_{h \in \mathcal{H}, \mathbf{f} \in \mathcal{F}_\epsilon} \hat{\mathcal{E}}(h, f_1, \cdots, f_T | \mathbf{Z})$.

The following theorem provides theoretical guarantee of our proposed regularizer's effectiveness.

Theorem 2 (Generalization Error)**.** Let $\delta > 0$ and $\mu_1, \mu_2, \ldots, \mu_T$ be the probability measure on $\mathcal{X} \times \mathbb{R}$. With probability of at least $1 - \delta$ in the draw of $\mathbf{Z} = (\mathbf{X}, \mathbf{Y}) \sim \prod_{t=1}^{T} \mu_t^n$, we have:

$$\begin{aligned} \mathcal{E}(\hat{h}, \hat{\mathbf{f}}) - \mathcal{E}(h^*, \mathbf{f}^*) \leq c_1 L \frac{G(\mathcal{H}(\mathbf{X}))}{nT} \\ + c_2 B \frac{\sqrt{\lambda_{min}^{-1}} \sup_h \|h(\mathbf{X})\|}{n\sqrt{nT}} + \sqrt{\frac{8 \ln{(4/\delta)}}{nT}} \end{aligned} \tag{10}$$

c_1, c_2 are universal constants, $G(\mathcal{H}(\mathbf{X}))$ is the Gaussian average defined as $G(\mathcal{H}(\mathbf{X})) = \mathbb{E}[\sup_{h \in \mathcal{H}} \sum_{kti} \gamma_{kti} h(\mathbf{x}_i^t) | \mathbf{x}_i^t]$, where $\{\gamma_{kti}\}$ is i.i.d standard normal variables. L is the Laplacian matrix of graph with T vertices and edge-weights $\{\omega_{ij}\}_{1\le i<j\le T}$, and λ_{min} is its smallest non-zero eigenvalue. B is any positive value that satisfies the condition $\sum_{i,j=1}^{T}\omega_{ij}\cdot dist^2(\nabla_A f_i, \nabla_A f_j) \le B^2$.

Proof. Please refer to the appendix for detailed proof. □

Some remarks over Theorem 2. **1).** The first term of the bound can be interpreted as the cost of estimating the shared representation learning function $h \in \mathcal{H}$. This term is typically of order $\frac{1}{n}$. The last term contains the confidence parameter. According to [32] the constant c_1 and c_2 are pretty large, so the last term typically makes limited contribution in the bound. **2).** The second or middle term contains the cost of estimating task-specific predictors $f \in \mathcal{F}$, and this term is typically of order $\frac{1}{\sqrt{n}}$. Here the positive constant B provides important insights into the relationship between our proposed regularizer and the error bound. The smaller our regularization term becomes, the smaller values B could take and in turn reduces the second term in the bound. In general, our generalization error result bounds the gap between the test error of the model trained from finite samples and that trained from infinite data, namely the theoretically optimal model/solution. In other words, Theorem 2 provides theoretical guarantee for our performance on actual test set.

4.2 Relation with Other MTL Frameworks

In this section, we mathematically elucidate the relation and difference between our proposed SRDML and other MTL methods with formal proof.

Natural generalization of shallow MTL. Following [51], traditional multi-task learning methods can be generally classified into two categories: multi-task feature learning and multi-task relation learning, with objective function $\min_{W,b,\Theta} L(W,b) + \lambda/2 \cdot tr(W^\top \Theta^{-1} W)$ and $\min_{W,b,\Sigma} L(W,b) + \lambda/2 \cdot tr(W^\top \Sigma^{-1} W)$, where Θ and Σ models the covariance between different features and tasks respectively. For any regularization-based Shallow MTL defined as above, it can be formulated as a *special case* under the general framework of SRDML, with identity feature extraction function h, linear task-specific function f and the corresponding regularizer on the input gradients.

Proof. Basically, when the feature extraction function h is identity function and each task-specific function f_t, $t = 1, 2, \cdots, T$ are linear functions, we know for any input $x \in \mathcal{X}$,

$$h(x) = x, \quad \nabla f_t(x) = w_t, \ \forall t \tag{11}$$

where w_t is the model parameter of linear model f_t. Hence, denote $W = [w_1; w_2; \cdots; w_T]$ and take the $dist()$ function in (3) to be inner product, by Lemma 1 we have $\sum_{i,j}\omega_{ij}\cdot dist(\nabla f_i(x), \nabla f_j(x)) = \sum_{i,j}\omega_{ij}\cdot\langle w_i, w_j\rangle = tr(W^\top \Omega W)$, where $\Omega = (\omega_{ij})$. Let Ω to be either Θ^{-1} or Σ^{-1} as in section 4.2 can finish the proof.

□

Relation with deep MTL. Define two hyperparameters: **1).** The coefficient of regularizer in SRDML λ and **2).** the number of layers ℓ before which the model is shared cross tasks. When λ equals 0 and ℓ is greater than 1 and less than L (total number of layers), SRDML degenerates to hard-parameter sharing. On the other hand, when ℓ equals to 1 and λ is greater than 0, our SRDML is equivalent to soft-parameter sharing.

Table 1: Attributes summary in CelebA and COCO.

T.Id	CelebA	COCO	T.Id	CelebA	COCO
1	ArchedEyebrows	person	11	PaleSkin	couch
2	BagsUnderEyes	cat	12	Sideburns	bed
3	BlackHair	dog	13	Smiling	dining table
4	BrownHair	horse	14	WavyHair	laptop
5	Chubby	car	15	WearingLipstick	tv
6	DoubleChin	truck	16	Young	cell phone
7	Goatee	bus	17		bottle
8	HeavyMakeup	motorcycle	18		cup
9	MouthSlightlyOpen	bicycle	19		bowl
10	Mustache	chair			

Proof. First, we define two hyperparameters:

- λ: The coefficient of our regularizer in SRDML
- ℓ: The index of the layer before which the model is shared cross different tasks.

Case 1. If $\lambda = 0$ and $1 < \ell < L$, where L (please differentiate this L with that for Lipschitz constant) denotes the total number of layers, our SRDML has no regularization and is simply equivalent to hard-parameter sharing.

Case 2. If $\lambda > 0$ and $\ell = 1$, each layer in our SRDML is separate for different tasks and the regularization is posed on all the layers, which is equivalent to soft-parameter sharing.

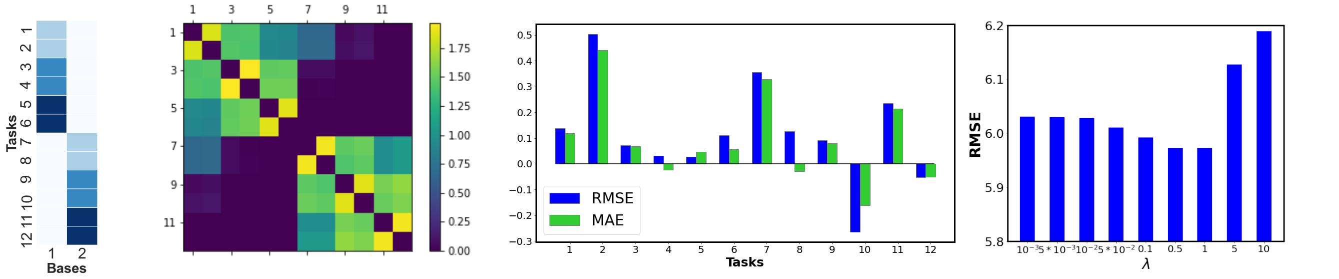
□

5 EXPERIMENTS

In this section, we validate SRDML on both synthetic and real-world datasets against state-of-the-art baselines, on various aspects including performance, sensitivity, qualitative analysis and ablation study. The experiments were performed on a 64-bit machine with 4-core Intel Xeon W-2123 @ 3.60GHz, 32GB memory and NVIDIA Quadro RTX 5000.

5.1 Experimental Settings

Controlled Synthetic Dataset. We first check the validity of SRDML on a controlled regression synthetic dataset. We generate T tasks ($T = 12$) and for each task i we generate m samples ($m = 100$). The input data $\mathbf{X}_i \in \mathbb{R}^{m\times d}$ ($d = 20$) for each task i is generated from $\mathbf{X}_i \sim \mathcal{N}(\eta_i, \mathbf{I})$ with mean vector η_i and identity covariance matrix $\mathbf{I}$. Next, we generate feature weight W by the following steps: **1)** Generate two base feature weights. As shown in Figure 4a, the first base feature weight (on the LHS column) corresponds to $\mathbf{w}_1 = (\mathbf{1}; \mathbf{0})^\top$ and the second base feature weight (on the RHS column) corresponds to $\mathbf{w}_2 = (\mathbf{0}; \mathbf{1})^\top$, where $\mathbf{1}$ and $\mathbf{0}$ each denotes a 10-dimensional all-one and all-zero vector respectively. In this way, $\mathbf{w}_1$ and $\mathbf{w}_2$ can simulate two different regions in the input X since the regions zeroed out by $\mathbf{w}$ will not be helpful in corresponding tasks. **2)** Generate task specific feature weight. Based on $\mathbf{w}_1$ and $\mathbf{w}_2$, we further consider creating different levels of saliency by multiplying the base feature weights by some magnitude parameter. Here we select 3 different magnitude parameters to create different levels of saliency for each base feature weight, and for each level of saliency we create two tasks which are basically twin tasks. For example, in Figure 4a, task 1 and task 2 are twin tasks which share the same level of saliency, and the lightest blue color means they are generated by the lowest magnitude parameter. We denote each generated task-specific feature weight as w_i, $i \in \{1, 2, \cdots, T\}$. The aforementioned logistics are basically symmetric for $\mathbf{w}_1$ and



(a) True W **(b) SRDML task relation** **(c) Per-task performance gain over STL.** **(d) sensitivity analysis.**

Figure 4: Results of synthetic dataset experiment. (a): Ground-truth of each task's feature weight. (b): Task relation learned by our proposed SRDML. Tasks from different bases show strong independency (as in dark purple), tasks from the same bases show clear similarities (as in light green), and each pair of twin tasks show very strong similarities (as in yellow). (c): The performance improvement of SRDML over single task learning in RMSE (blue bar) and MAE (green bar). As shown, SRDML model generally outperforms STL on the synthetic dataset by a large margin. (d): Sensitivity analysis on regularization coefficient.

$\mathbf{w}_2$. **3)** Add noise and create labels. We first inject some noise into each task's feature weight by randomly flipping the sign of the value in some positions of each w_i. The proportion of the flipped positions is controlled to guarantee the overall pattern can be well kept. Then, we generate the label for each task by $\mathbf{Y}_i = \mathbf{X}_i \cdot w_i + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(\mathbf{0}, 0.1 \cdot \mathbf{I})$ is random normal noise.

Real-world Dataset. We evaluate the proposed method on 3 real-world benchmarks with varying number of tasks and difficulty, including: multi-task version of CIFAR-10 [21] (**CIFAR-MTL**), a modified version of **CelebA** [27] and a modified version of **MS-COCO** [26]. To follow our model's assumption, all tasks are image classification ones. For CIFAR-MTL, we follow existing work [36] creating one task for each of the 10 classes in origianl CIFAR-10 dataset. There are 10 binary classification tasks with 2k training samples and 1k testing samples per task. CelebA has 200 thousand images of celebrity faces and each image is labeled with 40 facial attributes. We follow existing work [54] selecting 16 attributes more related to face appearance and ignore attributes around decoration such as eyeglasses and hat for our experiments. We randomly selected 30k training samples and include whole validation and test set. For MS-COCO we select 19 types of objects and remove those with too sparse labels. We include all images that contain at least two of the 19 types of objects and randomly split them into training and testing set by half. All results are reported on the test set. For hyperparameter tuning of our method, without further specification, we applied grid search on the range of $\{10^{-3}, 5 * 10^{-3}, \cdots, 0.5, 1\}$ for the regularization coefficient.

Comparison Methods We compare SRDML with various existing methods, including two baselines, three shallow and five deep sate-of-the-art MTL methods:

- **Practical Baselines:** *1). Single Task Learning (STL)* is to train a separate predictor for each task independently. *2) Hard Parameter Sharing (Hard-Share)* considers a shared representation learning backbone (e.g., convolutional layers in CNN) and task-specific prediction head.
- **Shallow MTL Methods:** *1) Lasso* [46] is an ℓ_1-norm regularized method which introduce sparsity into the model to reduce model complexity and feature learning, and that the parameter controlling the sparsity is shared among all tasks. *2) Joint Feature Learning (L_{21})* [14] assumes the tasks share a set of common features that represent the relatedness of multiple tasks. *3) Robust Multi-task Learning (RMTL)* [10] method assumes that some tasks are more relevant than others. It assumes that the model W can be decomposed into a low rank structure L that captures task-relatedness and a group-sparse structure S that detects outliers.
- **Deep MTL Methods:** *Multilinear Relationship Networks (MRNs)* places a tensor normal prior on task-specific layers of the deep multi-task learning model [28]. *2) Multi-gate Mixture-of-Experts (MMoE)* [30] adapt the Mixture-ofExperts (MoE) structure to multi-task learning by sharing the expert submodels across all tasks, while also having a gating network. *3) Progressive layered extraction (PLE)* [45] separates shared components and task-specific components explicitly and adopts a progressive routing mechanism to extract and separate deeper semantic knowledge gradually, improving efficiency of joint representation learning and information routing across tasks in a general setup. *4) Multi-task Learning as Multi-Objective Optimization (MGDA-UB)* [39] considers multi-task learning from optimization perspective by using Pareto optimality and Multiple Gradient Descent Algorithm. *5) Gradient Surgery for Multi-task Learning (PCGrad)* [49] aims to solve the problem of gradient interference by gradient surgery, which is basically by gradient projection to make sure the gradients of different tasks have direction smaller than 90°.

Implementation Details. All shallow MTL methods are implemented according to standard package MALSAR [56]. Deep MTL methods and our SRDML are built based on VGG-16 [41], which is a very popular architecture in computer vision. The convolutional layers are followed by one fully connected layer with 128 hidden neurons and one classification layer for our SRDML. Each model is trained by Adam [20]. For PCGrad, due to the fact that it is a gradient surgery method which is model-agnostic and can be applied onto any deep MTL method, we report its performance by combining it with the best baseline on each real-world dataset (i.e., Hard-Share on CIFAR-MTL, MGDA-UB on CelebA, MRN on COCO). In addition, we also consider applying PCGrad onto our own method SRDML, resulting in two versions of our method, namely SRDML and SRDML with PCGrad.

Table 2: Performance (%) on real-world large-scale multi-task learning datasets. Our proposed SRMTL outperforms most comparison methods on all three datasets. Bold and underline score are for the best and second best methods, respectively.

Model	CIFAR-MTL				CelebA				COCO			
	Accuracy	AUC	Precision	Recall	Accuracy	AUC	Precision	Recall	Accuracy	AUC	Precision	Recall
STL (Deep)	94.35	66.69	74,32	69.83	89.42	90.96	70.46	60.47	85.18	63.14	32.53	27.14
Hard-Share	94.70	95.56	76.30	72.28	89.24	91.38	71.40	58.84	85.11	73.68	**37.43**	19.84
Lasso	91.48	86.64	68.90	24.74	76.55	66.69	37.38	36.62	78.36	64.40	28.53	28.61
L21	91.50	87.58	68.01	29.32	76.09	66.12	37.11	36.13	75.07	65.02	28.95	27.34
RMTL	92.28	85.65	61.54	28.15	75.52	66.99	37.48	36.74	76.87	65.01	29.28	28.43
MRN	94.51	96.67	79.94	76.95	89.35	91.54	71.51	64.64	85.13	75.88	32.73	25.89
MMoE	93.53	93.17	73.42	69.32	77.57	67.84	68.79	58.92	81.20	62.37	33.08	26.14
PLE	94.01	93.32	75.26	70.15	83.21	69.32	70.03	59.72	82.53	63.42	35.27	27.53
MGDA-UB	90.74	84.38	57.80	24.10	90.03	92.92	73.42	62.65	84.51	73.68	36.17	16.08
PCGrad	95.11	96.69	79.03	74.82	90.11	92.87	73.51	62.92	85.42	74.39	34.52	25.26
SRDML	95.82	96.43	81.22	75.93	90.15	92.95	73.87	64.91	85.68	76.77	35.82	28.73
SRDML (w/. PCGrad)	**96.03**	**96.72**	**82.59**	**77.01**	**90.26**	**93.01**	**73.93**	**65.30**	**85.87**	**78.38**	36.14	**30.02**

5.2 Experimental Results

Effectiveness on controlled synthetic dataset. The empirical results on the regression synthetic dataset demonstrate that our model can generally outperform single task learning and is capable to capture the ground-truth task relations. Quantitative evaluation in Figure 4c shows that SRDML can outperform single task learning in general, which can be attributed to the effective knowledge sharing between task-specific layers. In addition, the task relationship pattern (i.e., w_{ij} in (4)) learned by SRDML as shown in Figure 4b is accurate and reasonable, since tasks belong to different bases are well-separated and meanwhile each pair of twin tasks shows very strong correlation (corresponds to those yellow boxes). Within each base, different pairs of twin tasks also show relatively strong relationship due to the fact that they share the same base and only differ in the magnitude.

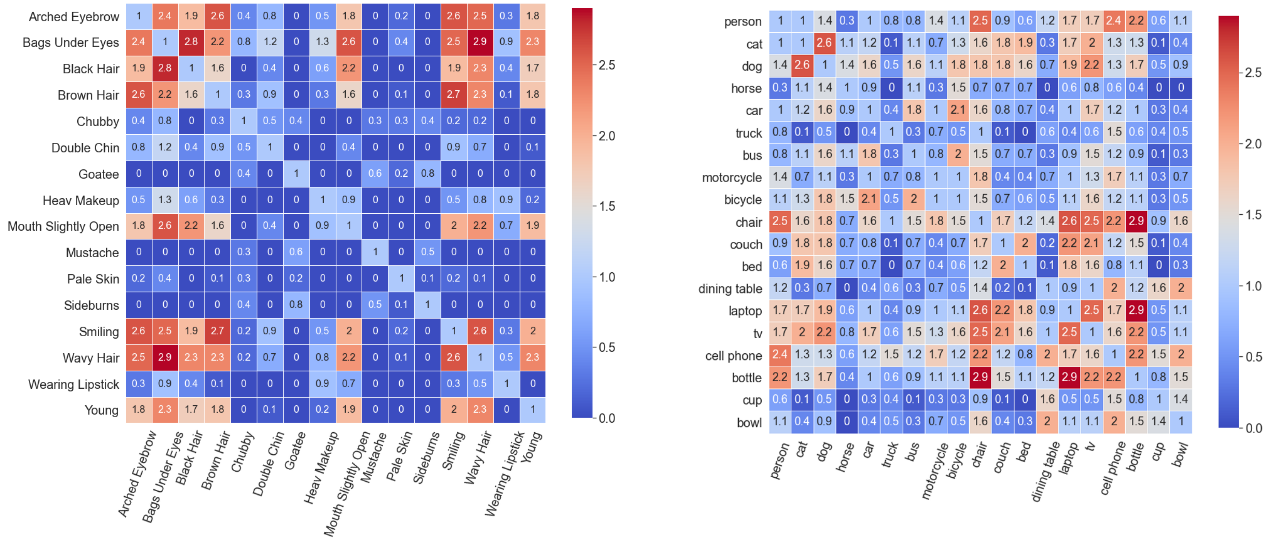
Sensitivity analysis. The sensitivity of hyperparameter λ in SRDML on synthetic dataset is shown in Figure 4d. As can be seen, the optimal value for λ is around 0.5 meansured by RMSE. The general "U" shape is potentially reasonable because as λ goes to infinity the regularization term would dominate the overall objective function while too small λ will reduce the functionality of the regularizer and finally degenerate to single task learning.

Effectiveness on real-world datasets.

- **CIFAR-MTL:** Table 2 shows the performance results of our proposed SRDML and other baselines on CIFAR-MTL dataset. We can make the following observations from the results. **1).** Deep multi-task learning models generally outperform shallow ones by a great margin, which confirms the importance of learning the deep representation features as well as the shared policy of feature extraction part which allows knowledge transfer across tasks. **2).** Our proposed SRDML outperforms baselines in majority of metrics and achieves comparable performance in rest. **3).** Combining with PCGrad can further improve the performance of SRDML due to the mitigated negative transfer by gradient surgery of PCGrad.
- **CelebA:** In this case, we tackle a larger and more challenging benchmark, where we tailored the dataset to contain 16 binary classification tasks with each one corresponding to a certain human's facial feature. As shown in Table 2, our model outperforms all comparison methods in majority of metrics, which is attributed to the potential fact that the salient regions in some tasks are close to those in the related tasks. For example, there are two tasks to classify whether a celebrity's beard is goatee or mustache, respectively. For both tasks the salient regions are highly overlapped around the mouth area (as can be seen in Section "Saliency map visualization" in appendix) so enforcing similar input gradients around the mouth area could improve the knowledge transfer and achieve better performance.
- **COCO:** To evaluate our model under various settings, we consider COCO which contains different types of objects like human, animals, vehicles, furniture, etc, and each type object has varying rate of occurrence. In Table 2, we report the task-average classification error with lower values indicating better performance. As shown in Table 2, our proposed SRDML outperforms all the baselines by a great margin. This experiment also validates the effectiveness of our model when the number of tasks is relatively large and the image context is complicated. Moreover, MMoE and PLE perform generally not quite well probably due to the fact that these two approaches are designed for multi-task learning under recommender system scenario, which is not similar to that in multi-task image classification, e.g., the number of tasks in our case is much larger and hence more challenging.

Qualitative analysis. Here we demonstrate that SRDML can learn reasonable task relations on challenging real-world dataset, by visualizing the task weight matrix (i.e., w_{ij} in (4)). As shown in Figure 5, many highlighted task relations are intuitive. In CelebA, our proposed SRDML successfully learned the similarity of tasks sharing the same/similar regions around face, lie "Arched Eyebrow" and "Bags Under Eyes"; "Black Hair", "Brown Hair" and "Wavy Hair"; "Goatee", "Sideburns" and "mustache", etc. On the other hand, our model can also learn reasonable task similarities in COCO, including "cat" and "dog"; "car", "bus" and "bicycle"; "couch" and "bed", etc.

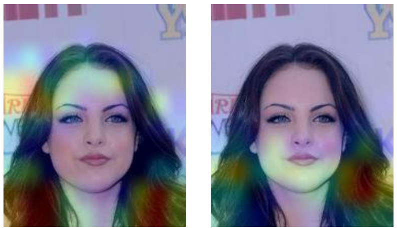
We also conduct qualitative analysis experiment on the saliency map generated by our proposed SRDML on similar or related tasks. As can be seen in Figure 6 and Figure 7, our proposed SRDML can generate saliency map focusing on similar regions for related



(a) CelebA

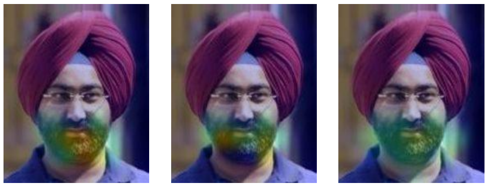
(b) COCO

Figure 5: Visualization of task relation learned by SRDML on real-world dataset. Zoom in for detail.



(a) black hair (b) brown hair

Figure 6: Saliency map generated by SRDML for hair tasks.



(a) goatee (b) mustache (c) no beard

Figure 7: Saliency map generated by SRDML for beard tasks.

tasks. For example, the saliency map generated for "Black hair" and "Brown hair" both generally overlap around the hair region of the woman, and the saliency map generated for three types of beard all overlap around the mouth and beard region of the man. Notice that the quality of saliency itself is not the main focus of this paper, but instead we are more interested in the task relation induced by the saliency map similarity (i.e., saliency across tasks).

Adaptive Regularizer on Contradicting Tasks. In this section, we conducted another sensitivity analysis when all tasks compete (we generate such synthetic dataset by following similar procedure introduced in Section 5.1), and the results in Table 3 demonstrate the efficacy of our regularization term which can adaptively decrease the task-similarity weight to zero and avoid competition.

Table 3: Sensitivity analysis on regularizer coefficient when tasks are contradicting. Our regularizer coefficient can adaptively reduce to zero and avoid negative transfer.

λ	1	0.1	0.01	0.001	0
RMSE.	2.726	1.550	1.405	1.393	1.392
MAE.	2.198	1.260	1.127	1.127	1.126

Ablation study. In this section, we present ablation study on the task relation learning part in the regularizer. Specifically, we remove the $\{\omega_{ij}\}_{1\leq i<j\leq T}$ in (3) and the coefficient for each term in the regularizer is just the hyperparameter λ. We conduct experiments on all three real-world dataset to see the difference and the results are shown in Table 4.

Table 4: Ablation study on adaptive regularizer (Accuracy)

	CIFAR-MTL	CelebA	MS-COCO
Ours. (**w/o** regularizer)	94.92	89.74	85.18
Ours. (**w/.** regularizer)	**95.82**	**90.15**	**85.68**

6 CONCLUSION

Learning interpretable task relation is challenging in multi-task learning problem. In this paper, we proposed Saliency-regularized Deep Multi-task Learning (SRDML) framework which regularizes the input gradient of different tasks by saliency and achieves good task relation interpretability. Instead of regularizing parameters like existing work, we directly regularize in functional space which allows better expressiveness. Theoretical analyses show that our regularizer can help reduce the generalization error. Experiments on multiple synthetic and real-world datasets demonstrate the effectiveness and efficiency of our methods in various metrics, compared with several comparison methods and baselines. The reasonableness of the task relation learned by SRDML is also validated on different challenging real-world datasets.

REFERENCES

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

[2] Andreas Argyriou, Charles A Micchelli, and Massimiliano Pontil. 2009. When is there a representer theorem? Vector versus matrix regularizers. *The Journal of Machine Learning Research* 10 (2009), 2507–2529.

[3] Andreas Argyriou, Charles A Micchelli, Massimiliano Pontil, and Yiming Ying. 2007. A spectral regularization framework for multi-task structure learning.. In *NIPS*, Vol. 1290. Citeseer, 1296.

[4] Jonathan Baxter. 2000. A model of inductive bias learning. *Journal of artificial intelligence research* 12 (2000), 149–198.

[5] Shai Ben-David, Johannes Gehrke, and Reba Schuller. 2002. A theoretical framework for learning from a pool of disparate data sources. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 443–449.

[6] Shai Ben-David and Reba Schuller. 2003. Exploiting task relatedness for multiple task learning. In *Learning theory and kernel machines*. Springer, 567–580.

[7] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.

[8] Christopher M Bishop. 2006. *Pattern recognition and machine learning*. springer.

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

[10] Jianhui Chen, Jiayu Zhou, and Jieping Ye. 2011. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 42–50.

[11] Michael Crawshaw. 2020. Multi-Task Learning with Deep Neural Networks: A Survey. *arXiv preprint arXiv:2009.09796* (2020).

[12] 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 (volume 2: short papers)*. 845–850.

[13] Thomas Elsken, Jan Hendrik Metzen, Frank Hutter, et al. 2019. Neural architecture search: A survey. *J. Mach. Learn. Res.* 20, 55 (2019), 1–21.

[14] An Evgeniou and Massimiliano Pontil. 2007. Multi-task feature learning. *Advances in neural information processing systems* 19 (2007), 41.

[15] Theodoros Evgeniou, Charles A Micchelli, Massimiliano Pontil, and John Shawe-Taylor. 2005. Learning multiple tasks with kernel methods. *Journal of machine learning research* 6, 4 (2005).

[16] Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 109–117.

[17] Stas Goferman, Lihi Zelnik-Manor, and Ayellet Tal. 2011. Context-aware saliency detection. *IEEE transactions on pattern analysis and machine intelligence* 34, 10 (2011), 1915–1926.

[18] Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. 2020. Learning to branch for multi-task learning. In *International Conference on Machine Learning*. PMLR, 3854–3863.

[19] Laurent Jacob, Francis Bach, and Jean-Philippe Vert. 2008. Clustered multi-task learning: A convex formulation. *arXiv preprint arXiv:0809.2085* (2008).

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

[21] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[22] Abhishek Kumar and Hal Daume III. 2012. Learning task grouping and overlap in multi-task learning. *arXiv preprint arXiv:1206.6417* (2012).

[23] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

[24] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066* (2015).

[25] Min Lin, Qiang Chen, and Shuicheng Yan. 2013. Network in network. *arXiv preprint arXiv:1312.4400* (2013).

[26] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

[27] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[28] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and S Yu Philip. 2017. Learning Multiple Tasks with Multilinear Relationship Networks. In *NIPS*.

[29] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. 2017. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5334–5343.

[30] 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*. 1930–1939.

[31] Aravindh Mahendran and Andrea Vedaldi. 2016. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision* 120, 3 (2016), 233–255.

[32] Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. 2016. The benefit of multitask representation learning. *Journal of Machine Learning Research* 17, 81 (2016), 1–32.

[33] 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.

[34] Wanli Ouyang, Xiao Chu, and Xiaogang Wang. 2014. Multi-source deep learning for human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2329–2336.

[35] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*. 1085–1097.

[36] Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. 2017. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *arXiv preprint arXiv:1711.01239* (2017).

[37] Ueli Rutishauser, Dirk Walther, Christof Koch, and Pietro Perona. 2004. Is bottom-up attention useful for object recognition?. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, Vol. 2. IEEE, II–II.

[38] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*. 618–626.

[39] Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. *arXiv preprint arXiv:1810.04650* (2018).

[40] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).

[41] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[42] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. 2014. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806* (2014).

[43] Gjorgji Strezoski, Nanne van Noord, and Marcel Worring. 2019. Learning task relatedness in multi-task learning for images in context. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*. 78–86.

[44] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. 2019. Adashare: Learning what to share for efficient deep multi-task learning. *arXiv preprint arXiv:1911.12423* (2019).

[45] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Fourteenth ACM Conference on Recommender Systems*. 269–278.

[46] Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* 58, 1 (1996), 267–288.

[47] Sen Wu, Hongyang R Zhang, and Christopher Ré. 2020. Understanding and improving information transfer in multi-task learning. *arXiv preprint arXiv:2005.00944* (2020).

[48] Yongxin Yang and Timothy M Hospedales. 2016. Trace norm regularised deep multi-task learning. *arXiv preprint arXiv:1606.04038* (2016).

[49] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782* (2020).

[50] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*. Springer, 818–833.

[51] Yu Zhang and Qiang Yang. 2021. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering* (2021).

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

[53] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. 2014. Facial landmark detection by deep multi-task learning. In *European conference on computer vision*. Springer, 94–108.

[54] Xiangyun Zhao, Haoxiang Li, Xiaohui Shen, Xiaodan Liang, and Ying Wu. 2018. A modulation module for multi-task learning with applications in image retrieval. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 401–416.

[55] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2921–2929.

[56] Jiayu Zhou, Jianhui Chen, and Jieping Ye. 2011. Malsar: Multi-task learning via structural regularization. *Arizona State University* 21 (2011).

Appendix

A PROOF OF THEOREM 2

In this section, we provide the proof of our model's generalization error bound. First, we introduce some definitions and lemmas which will be continuously used, and at the end of this section we present the proof for Theorem 2.

In general, we will use γ to denote a generic vector of i.i.d standard normal variables, whose dimension will be clear in context. In addition, without further specification, we will use K, T, n to denote the (flattened) dimension of the output space from the feature extraction function h, number of tasks, and number of training samples, respectively. We denote the representation class for task-specific function f and representation extraction function h as $\mathcal{F}$ and $\mathcal{H}$, respectively. Two hypothesis classes here can be very general, and the only assumption here is that $\forall f \in \mathcal{F}$, f has Lipschitz constant at most L, for any positive L.

Definition 3. Given a set $V \subseteq \mathbb{R}^n$, define the Gaussian average of V as:

$$G(V) = \mathbb{E} \sup_{v \in V} \langle \gamma, v \rangle = \mathbb{E} \sup_{v \in V} \sum_{i=1}^{n} \gamma_i v_i \tag{12}$$

As mentioned in section 3.1 in main paper, we denote the feature representation learning part as function $h \in \mathcal{H}$. As we will see later, the complexity of representation class $\mathcal{H}$ is important in our proof for the error bound, so we define a measure of its complexity by Gaussian average.

Definition 4. Given observed input data $\mathbf{X} \in \mathcal{X}^{Tn}$, define a random set $\mathcal{H}(\mathbf{X}) \subseteq \mathbb{R}^{KTn}$ by

$$\mathcal{H}(\mathbf{X}) = \{(h_k(\mathbf{x}_i^t)) : h \in \mathcal{H}\}. \tag{13}$$

The Gaussian average over $\mathcal{H}(\mathbf{X})$ can be defined accordingly as

$$G(\mathcal{H}(\mathbf{X})) = \mathbb{E}[\sup_{h \in \mathcal{H}} \sum_{kti}^{K,T,n} \gamma_{kti} h_k(\mathbf{x}_{ti}) | \mathbf{x}_{ti}] \tag{14}$$

The following lemmas are useful in our proof later, and we introduce them here in advance.

Lemma 1. $\forall A, C \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{m \times m}$,

$$tr(A^\top BC) = \sum_{i,j}^{m} B_{ij} \sum_{k=1}^{n} A_{ik} C_{jk}. \tag{15}$$

Lemma 2. Suppose $\mathcal{X} \subseteq \mathbb{R}^K$ is an open set, and two differentiable functions $f_1, f_2 \ : \mathcal{X} \to \mathbb{R}$. $\forall x \in \mathcal{X}$, if $\exists B > 0$, $s.t.$ $\|\nabla f_1(x) - \nabla f_2(x)\| < B$, then

$$\lim_{\Delta x \to 0} | \frac{f_1(x + \Delta x) - f_1(x)}{\|\Delta x\|} - \frac{f_2(x + \Delta x) - f_2(x)}{\|\Delta x\|} | < B. \tag{16}$$

Given everything above, we can prove our Theorem 2 with Maurer et al. [32]. However, due to the limited space of appendix here, we decide to put the formal proof for our Theorem 2 along with the proof for everything above into the **supplementary materials**. Please refer to there for the formal proof. Thanks!

B DETAILS ON SYNTHETIC DATASET GENERATION

- "What are base feature weights": Since we want to generate tasks with different level of similarity in our synthetic dataset, we achieved it by controlling the similarity in the feature weight (i.e., w) of different tasks. The base feature weights w_1 and w_2 are basically two vectors (with length equal to number of features) for generating the feature weight vectors for all the tasks. We call them "base" feature weight because they serve as the base vector or unit vector for generating all the tasks' feature weights. In addition, w_1 and w_2 are orthogonal and each has length 1 in any dimension.
- "How are base feature weights used": The base feature weights are used to generate each task's feature weight in the following steps: 1) We choose which base the current task belongs to. In our setting, we chose the first half of tasks to belong to the first base (i.e., w_1) and the second half of tasks to belong to the second base (i.e., w_2). Since two bases are orthogonal, they can actually simulate two non-overlapping regions in pictures which means tasks from different bases should not be similar while those from the same base should be similar since, they focus on the same region. 2) Within each base, we multiply the base vector (i.e., w_1 and w_2) by some positive integers to generate the actual feature weight for the tasks. For example, we multiply w_1 by integer 1, 2 and 3 to generate the feature weight vectors for the first half of tasks.

C EFFECT OF NORMALIZATION ON INPUT GRADIENT

We have added an experiment on our method with normalizing the input gradients and compare its results with our original method (e.g., without normalization in input gradients) on ALL 3 real-world dataset we used in our original paper. As shown in the Table 5, adding normalization did not obviously change the performance in task-average classification error. The task-average classification error decreased by < 0.2% on CIFAR-MTL and increased by around 0.1% on CelebA and COCO. One explanation is, for similar tasks like "Black hair" and "Brown hair" in CelebA, we empirically observed that the magnitude for the gradients was close to each other, which might limit the point in applying gradient normalization in such case.

	CIFAR-MTL	CelebA	MS-COCO
SRDML **w/o** normalization	4.18	9.91	14.32
SRDML **w/** normalization	4.02	10.03	14.41

Table 5: Normalization of input gradient

D EXTRA REMARKS ON THEOREM 2

In this section, we provide more remarks on our main theorem, namely Theorem 2, for better understanding.

- The equation above bounds the gap between the test error of the model trained from finite samples and that trained from infinite data, namely the theoretically optimal model/solution. In other words, Theorem 2 provides theoretical guarantee for our performance on actual test error.
- In Equation 2-9, we assume all tasks share the same set of X which is a very common case in Multi-task Learning on image dataset. Theorem 2 does not need different tasks to have different X(t), since $\mu_1 = \mu_2 = \cdots = \mu_T = \mu$ is a special case of the version in Theorem 2. Our current assumption is actually a more general one and can handle the case in Equation 2-9.