Deep Safe Multi-Task Learning

Zhixiong Yue*, Feiyang Ye*, Yu Zhang, Christy Liang, and Ivor W. Tsang

Abstract—In recent years, Multi-Task Learning (MTL) has attracted much attention due to its good performance in many applications. However, many existing MTL models cannot guarantee that their performance is no worse than their single-task counterparts on each task. Though some works have empirically observed this phenomenon, little work aims to handle the resulting problem. In this paper, we formally define this phenomenon as negative sharing and define safe multi-task learning where no negative sharing occurs. To achieve safe multi-task learning, we propose a Deep Safe Multi-Task Learning (DSMTL) model with two learning strategies: individual learning and joint learning. We theoretically study the safeness of both learning strategies in the DSMTL model to show that the proposed methods can achieve some versions of safe multi-task learning. Moreover, to improve the scalability of the DSMTL model, we propose an extension, which automatically learns a compact architecture and empirically achieves safe multi-task learning. Extensive experiments on benchmark datasets verify the safeness of the proposed methods.

Index Terms—Multi-task learning, Negative sharing, Safe multi-task learning

1 INTRODUCTION

MUlti-Task Learning (MTL) [5], [59], which aims to improve the generalization performance of multiple learning tasks by sharing knowledge among those tasks, has attracted much attention in recent years. Compared with single-task learning that learns each task independently, MTL not only improves the performance for some or all the tasks but also reduces the training and inference time. Therefore, MTL has been widely used in many Computer Vision (CV) applications, such as human action recognition [28], face attribute estimation [18], age estimation [61], and dense prediction tasks [53].

Although MTL has demonstrated its usefulness in many applications, MTL cannot guarantee to improve the performance of all the tasks compared with single-task learning. Specifically, as empirically observed in [15], [24], [45], [48], [49], when learning multiple tasks together, many existing MTL models can achieve better performance on some tasks than their single-task counterparts but underperform on the other tasks. Such phenomenon is called the negative sharing phenomenon in this paper, which is similar to the ‘negative transfer’ phenomenon [54] in transfer learning [55] but with some differences as discussed later. One reason for the occurrence of negative sharing is that there are partially related or even unrelated tasks among tasks under the investigation, making jointly learning those tasks impair the performance of some tasks.

To the best of our knowledge, there is little work to study the negative sharing problem for MTL. To fill this gap, in this paper, we firstly give a formal definition for negative sharing that could occur in MTL. Then we formally define an ideal and also basic situation for MTL called safe multi-task learning, where the generalization performance of an MTL model is no worse than its single-task counterpart on each task. That is, there is no negative sharing occurred. According to the definition of MTL [5], [59], we can see that every MTL model is required to achieve safe multi-task learning. Otherwise, single-task learning is more preferred than MTL, since an unsafe MTL model may bring the risk of worsening the generalization performance of some or even all the tasks. As true data distributions in multiple tasks are usually unknown so that safe multi-task learning is hardly to measure, we formally define empirically safe multi-task learning and probably safe multi-task learning, which are measurable.

To achieve empirically/probably safe multi-task learning, we propose a Deep Safe Multi-Task Learning (DSMTL) model whose architecture consists of a public encoder shared by all the tasks and a private encoder for each task. The public encoder and a private encoder of a task are combined via a gating mechanism to form the entire encoder for that task. To train the DSMTL model, we propose two learning strategies: individual learning (denoted by DSMTL-IL) and joint learning (denoted by DSMTL-JL), which learn model parameters separately and jointly, respectively. For those two strategies, we provide theoretical analyses to show that they can achieve some versions of both empirically safe multi-task learning and probably safe multi-task learning.

To improve the scalability of the DSMTL model with respect to the number of tasks, we propose an extension called DSMTL with Architecture Learning (DSMTL-AL), which leverages neural architecture search to learn a more compact architecture with fine-grained modular splitting. Specifically, we allow the DSMTL-AL model to learn where to switch to the private encoder while forwarding in the public encoder. In this way, the DSMTL-AL model can save the first few modules in the private encoders and hence improve the scalability.

Extensive experiments on benchmark datasets, including CityScapes, NYUv2, PASCAL-Context, and Taskonomy, demonstrate the effectiveness of the proposed DSMTL-IL,
DSMTL-JL, and DSMTL-AL methods.

The main contributions of this paper are summarized as follows.

- We provide formal definitions for MTL, including negative sharing, safe multi-task learning, empirically safe multi-task learning, and probably safe multi-task learning.
- We propose the simple and effective DSMTL model with two learning strategies, which is guaranteed to achieve some versions of empirically/probably safe multi-task learning.
- We propose the DSMTL-AL method, which is an extension of the DSMTL methods, to learn a compact architecture with good scalability.
- Extensive experiments demonstrate that empirically the proposed methods can achieve safe multi-task learning and that they outperform state-of-the-art baseline models.

2 \textbf{Related Work}

MTL has been extensively studied in recent years \cite{10, 17, 22, 58, 60}. How to design a good network architecture for MTL is an important issue. The most widely used architecture is the multi-head hard sharing architecture \cite{5}, \cite{32}, \cite{43}, which shares the first several layers among all the tasks and allows the subsequent layers to be specific to different tasks. Then, to better handle task relationships, different MTL architectures have been proposed. For example, \cite{38} proposes a cross-stitch network to learn to linearly combine hidden representations of different tasks. \cite{34} proposes a multi-gate mixture-of-experts model which adopts the mixture-of-experts model by sharing expert submodels across all tasks, while having a gating network trained to optimize each task. \cite{31} proposes a Multi-Task Attention Network (MTAN), which consists of a shared network and an attention module for each task so that both shared and private feature representations can be learned via the attention mechanism. \cite{13} proposes a Neural Discriminative Dimensionality Reduction (NDDR) layer to enable automatic feature fusion at every layer for different tasks. \cite{48} proposes an Adaptive Sharing (AdaShare) method to learn the sharing pattern through a policy that selectively chooses which layers to be executed for each task. \cite{9} proposes an Adaptive Feature Aggregation (AFA) layer, where a dynamic aggregation mechanism is designed to allow each task to adaptively determine the degree of the knowledge sharing between tasks. PS-MCNN \cite{4} adopts both shared network and task-specific network by performing the concatenation operation after each block to learn shared and task-specific representations. PLE \cite{49} separates shared components and task-specific components explicitly and adopts a progressive routing mechanism to extract semantic knowledge gradually for MTL. Some routing-based methods are proposed, including Multi-Agent Reinforcement Learning (MARL) \cite{42} that allows the MTL network to dynamically self-organize its architecture in response to the input, Stochastic Filter Groups (SFG) \cite{2} that assigns convolution kernels in each layer to the “specialist” or “generalist” group, and Task Routing Layer (TRL) \cite{46} that allows for a single model to fit to many tasks within its parameter space with task-specific masking.

Instead of hand-crafting architectures for MTL, there are some works to leverage techniques in Neural Architecture Search (NAS) \cite{30} to automatically search MTL architectures with good performance. For example, \cite{33} dynamically widens a multi-layer network to create a tree-like deep architecture, where similar tasks reside in the same branch. \cite{27} proposes an evolutionary architecture search algorithm to search blueprints and modules that are assembled into an MTL network. \cite{12} searches inter-task layers for better feature fusion across tasks. \cite{16} proposes a differentiable architecture search algorithm to learn branching blocks to construct a tree-structured neural network for MTL. \cite{3} automatically determines the branching architecture for the encoder in a multi-task neural network under resource constraints. \cite{15} aims to learn where to share or branch within a network for multiple tasks. \cite{47} designs a task switching network that can learn to switch between tasks with a constant number of parameters which is independent of the number of tasks. All the aforementioned works do not study how to achieve safe multi-task learning, which is the focus of this paper.

The safeness of machine learning methods has drawn attention in recent years \cite{14}, \cite{25}, \cite{26}, \cite{50}, \cite{51}, \cite{52}. \cite{26} proposes a safe semi-supervised support vector machine that performs no worse than the supervised counterpart, leading to the safeness in the use of unlabeled data. \cite{25} addresses the safe weakly supervised learning problem by integrating multiple weakly supervised learners, which is guaranteed to derive a safe prediction under a mild condition. \cite{14} proposes a deep semi-supervised learning method to alleviate the harm caused by class distribution mismatch. Moreover, there are some works \cite{50}, \cite{51}, \cite{52} to address the safeness in multi-view clustering. \cite{52} proposes reliable multi-view clustering, which empirically performs no worse than its single-view counterpart and proves that its performance will not significantly degrade under some assumptions. \cite{51} proposes deep safe multi-view clustering to reduce the risk of performance degradation caused by view increasing and hence to guarantee to achieve the safeness in multi-view clustering. \cite{50} proposes a bi-level optimization framework to achieve safe incomplete multi-view clustering.

Different from the aforementioned works that address the safeness in semi-supervised learning, weakly supervised learning, and multi-view clustering, our work focuses on the safeness in multi-task learning.

3 \textbf{Definitions}

In this section, we formally introduce some definitions to measure the safeness in MTL.

We first define the negative sharing phenomena.

\textbf{Definition 1} (Negative Sharing). For an MTL model which is trained on multiple learning tasks jointly, if its generalization performance on some tasks is inferior to the generalization performance of the corresponding single-task counterpart that is trained on each task separately, then negative sharing occurs.

\textbf{Remark 1}. Negative sharing could occur when some tasks are partially or totally unrelated to other tasks. In this case, manually
enforcing all the tasks to have some forms of sharing will impair the performance of some or even all the tasks. In Definition 1, the MTL model and its single-task counterpart usually have similar architectures, since totally different architectures could bring additional confounding factors. Moreover, negative sharing is similar to negative transfer [54] in transfer learning [55]. However, knowledge transfer in transfer learning is directed as it is from a source domain to a target domain, while knowledge sharing in MTL is among all the tasks, making it usually undirected.

Definition 2 (Safe Multi-Task Learning). When negative sharing does not occur for an MTL model on a dataset, this MTL model is said to achieve safe multi-task learning on this dataset.

Remark 2. Safe multi-task learning is an ideal situation for an MTL model to achieve. However, the generalization performance is hard to evaluate during the learning process, so it is hard to determine whether an MTL model can achieve safe multi-task learning.

As the empirical/training loss is easy to compute during the learning process, we present the following definition based on the empirical loss to measure the empirical safety of MTL models.

Definition 3 (Empirically Safe Multi-Task Learning). (1) If the empirical loss of an MTL model on each task is no larger than that of its single-task counterpart, this MTL model is said to achieve empirically individual safe multi-task learning. (2) If the average of empirical losses of an MTL model on all the tasks is no larger than that of its single-task counterpart, this MTL model is said to achieve empirically average safe multi-task learning.

Remark 3. In Definition 3 we define two versions of empirically safe multi-task learning. It is easy to see that an MTL model satisfying empirically individual safe multi-task learning can achieve empirically average safe multi-task learning but not vice versa, which indicates that empirically individual safe multi-task learning is weaker than empirically individual safe multi-task learning. Even though empirically safe multi-task learning is easy to measure based on the empirical loss of each task, an MTL model that achieves empirically safe multi-task learning cannot have guarantee to achieve safe multi-task learning, since there is a gap between the empirical loss and the expected loss that is to measure the generalization performance. Hence, empirically safe multi-task learning is a loose version of safe multi-task learning.

With m learning tasks in an MTL problem, \( E_t \) and \( E_{STL}^t \) denote the expected losses of an MTL model and its single-task counterpart on task \( t \), respectively. The corresponding average expected losses are denoted by \( E \) and \( E_{STL} \), respectively. \( n \) denotes the average number of samples in all the tasks. With the above notations, we can define probably safe multi-task learning as follows.

Definition 4 (Probably Safe Multi-Task Learning). (1) For an MTL model trained on \( m \) tasks, if for \( 0 < \delta < 1 \), there exist \( m \) constants \( \epsilon_t \geq 0 \) such that \( E_t + \epsilon_t \leq E_{STL}^t + \rho_n \) holds with at least probability \( 1 - \delta \) for any \( t \in [1, m] \), where \( \rho_n \) is a function of \( n \) satisfying \( \lim_{n \to +\infty} \rho_n = 0 \), then this MTL model is said to achieve probably individual safe multi-task learning.

(2) If for \( 0 < \delta < 1 \), there exists a constant \( \epsilon \geq 0 \) such that \( E + \epsilon \leq E_{STL} + \rho_n \) holds with at least probability \( 1 - \delta \), where \( \rho_n \) is a function of \( n \) satisfying \( \lim_{n \to +\infty} \rho_n = 0 \), then this MTL model is said to achieve probably average safe multi-task learning.

Remark 4. Different from empirically safe multi-task learning which is measured based on empirical losses, probably safe multi-task learning is based on expected losses, making it a tighter approximation of safe multi-task learning than empirically safe multi-task learning. Compared with safe multi-task learning, probably safe multi-task learning is easy to be measured based on some analysis tool as verified in Section 5. According to Definition 4, it is easy to see when \( n \) is large enough, probably individual safe multi-task learning could become safe multi-task learning in a large probability. Between the two versions of probably safe multi-task learning, similar to empirically safe multi-task learning, probably average safe multi-task learning is weaker.

As discussed above, empirically safe multi-task learning and probably safe multi-task learning are two measures for an MTL model to achieve but few works can guarantee to achieve that. In the next section, we will propose the DSMTL model with such guarantees under mild conditions.

4 DSMTL

In this section, we present the proposed DSMTL model. Beside the network architecture, we introduce two strategies to learn model parameters, leading to two variants (i.e., DSMTL-IL and DSMTL-JL).

![Fig. 1. An illustration for the architecture of the DSMTL model with three tasks (i.e., \( m = 3 \)). Here without loss of generality, we assume different tasks share the input data. For task \( t \), an input \( x \) is first fed into both the public encoder \( f_S \) and private encoder \( f_t \), then it goes through the gate \( g_t \) to obtain the combined feature representation, and finally it is through the private decoder \( h_t \) to obtain the output \( \hat{y}_t \).](image-url)
m gates \( \{g_t\}_{t=1}^m \) for \( m \) tasks, and \( m \) private decoders \( \{h_t\}_{t=1}^m \) for \( m \) tasks. For task \( t \), its model consists of the public encoder \( f_S \), the private encoder \( f_t \), the gate \( g_t \), and the private decoder \( h_t \), where \( f_S \) and \( f_t \) are combined by \( g_t \). Specifically, given a data sample \( x \), the gate \( g_t \) in task \( t \) receives two inputs: \( f_S(x) \) and \( f_t(x) \), and outputs \( g_t(f_S(x), f_t(x)) \), which is fed into \( h_t \) to obtain the final prediction \( h_t(g_t(f_S(x), f_t(x))) \), which is used to define a loss for \( x \). Here the public encoder \( f_S \) and private encoders \( \{f_t\}_{t=1}^m \) usually have the same network structure. The private decoders are designed to be task-specific as different tasks may have different types of loss functions.

Here the gate \( g_t \) is to determine the contributions of \( f_S \) and \( f_t \). Ideally, when task \( t \) is unrelated to other tasks, \( g_t \) should choose \( f_t \) only. On another extreme where all the tasks have the same data distribution, all the tasks should use the same model and hence \( g_t \) should choose \( f_S \) only. In cases between these two extremes, \( g_t \) can combine \( f_S \) and \( f_t \) in proportion. To achieve the aforementioned effects, we use a simple convex combination function for \( g_t \)

\[
g_t(f_S(x), f_t(x)) = \alpha_t f_S(x) + (1 - \alpha_t) f_t(x),
\]

where \( \alpha_t \in [0, 1] \) defines the weight of \( f_S(x) \) for task \( t \) and it is a learnable parameter. When \( \alpha_t \) equals 0, only the private encoder \( f_t \) will be used, which corresponds to the unrelated case. When \( \alpha_t \) is 1, only the public encoder \( f_S \) will be used, which corresponds to the case that all the tasks follow identical or similar distributions. When \( \alpha_t \) is between 0 and 1, \( f_S \) and \( f_t \) are combined with proportions \( \alpha_t \) and \( 1 - \alpha_t \), respectively, and \( \alpha_t \) can be adaptively learned to minimize the training loss on task \( t \).

The average empirical loss of all the tasks is defined as

\[
\frac{1}{mn} \sum_{t=1}^m \sum_{i=1}^n \mathcal{L}_t(y_i^t, h_t(g_t(f_S(x_i^t), f_t(x_i^t))));
\]

where \( x_i^t \) denotes the \( i \)th data point in task \( t \), \( y_i^t \) denotes the label of \( x_i^t \) in task \( t \), without loss of generality, different tasks are assumed to have the same number of data samples, which is denoted by \( n \), and \( \mathcal{L}_t \) denotes the loss function for task \( t \) (e.g., the pixel-wise cross-entropy loss for the semantic segmentation task, the \( L_1 \) loss for the depth estimation task, and the element-wise dot product loss for the surface normal prediction task). The DSMTL model is to minimize the average empirical loss in Eq. (2) to learn its model parameters. In the following sections, we provide two learning strategies (e.g., individual learning and joint learning) to learn model parameters, leading to two variants of DSMTL, including DSMTL with Individual Learning (DSMTL-IL) and DSMTL with Joint Learning (DSMTL-JL).

### 4.2 DSMTL-IL

The set of all the parameters in the DSMTL model is denoted by \( \Theta \). We divide \( \Theta \) into \( \Theta_S \) and \( \Theta_H \), where \( \Theta_S \) includes all the parameters in \( \{f_t\}_{t=1}^m \) and \( \{h_t\}_{t=1}^m \), and \( \Theta_H \) includes the parameters in \( f_S \) and \( \{g_t\}_{t=1}^m \). The individual learning strategy consists of two stages. The first stage is to optimize \( \Theta_H \) by fixing each \( \alpha_t \) to 0. Then by fixing the learned \( \Theta_H \) in the first stage, the second stage is to learn \( \Theta_H \). As the single-task model for each task consists of an encoder and a decoder whose structures are identical to \( f_t \) and \( h_t \), respectively, the first stage is equivalent to learning a single-task model for each task, and after that, the second stage can learn the shared encoder \( f_S \) and the gate \( \{g_t\}_{t=1}^m \). Formally, the objective function of the DSMTL-IL model is formulated as

\[
\min_{\Theta} \left[ \min_{\Theta_S} \frac{1}{mn} \sum_{t=1}^m \sum_{i=1}^n \mathcal{L}_t(y_i^t, h_t(g_t(f_S(x_i^t), f_t(x_i^t)))) \right].
\]

The DSMTL-IL model can be proved to achieve both empirically individual safe multi-task learning and probably individual safe multi-task learning as shown in Section 5.2 due to its two-stage optimization process.

### 4.3 DSMTL-JL

Different from the DSMTL-IL model which optimizes two partitions of model parameters sequentially, the DSMTL-JL model adopts a joint learning strategy to learn \( \Theta \) together. Formally, the objective function of the DSMTL-JL model is formulated as

\[
\min_{\Theta} \frac{1}{mn} \sum_{t=1}^m \sum_{i=1}^n \mathcal{L}_t(y_i^t, h_t(g_t(f_S(x_i^t), f_t(x_i^t)))).
\]

The joint learning strategy allows the DSMTL model to learn all the model parameters, which is more flexible for deep neural networks in an end-to-end learning manner. Different from the DSMTL-IL model, the DSMTL-JL model can achieve both empirically average safe multi-task learning and probably average safe multi-task learning as proven in Section 5.3.

### 5 Analyzes

In this section, we provide theoretical analyses to analyze the safeness of both the DSMTL-IL and DSMTL-JL methods.

#### 5.1 Preliminary

With \( m \) tasks, the probability measure for the data distribution in task \( t \) is denoted by \( \mu_t \) and the data in all the tasks take the form of \( (X, Y) \sim \prod_{t=1}^m (\mu_t)^n \), where \( X_t = (x_{1t}^1, ..., x_{mt}^n) \) denotes the data in task \( t \), \( X = (X_1, ..., X_m) \), and \( Y \) denotes labels for \( X \). Here we consider the encoders \( f_1, ..., f_m, f_S : X \to \mathbb{R}^q \) as mapping functions chosen from a hypothesis class \( F \) and the decoders \( h_1, ..., h_m \) as mapping functions chosen from a hypothesis class \( H \). To facilitate the analysis, we introduce following assumption.

**Assumption 1.** Assume that (i) \( \mathcal{L}_t(\cdot, \cdot) \in [0, 1] \) for \( t = 1, ..., m \) is 1-Lipschitz w.r.t the second argument; (ii) The hypothesis class \( F \) is uniformly bounded; (iii) The functions in hypothesis class \( H \) are Lipschitz continuous; (iv) \( 0 \in F \) and \( h(0) = 0 \) holds for all the functions \( h \) in \( H \).

To analyze the safeness of the DSMTL variants, we compare with the corresponding Single-Task Learning (STL) model, which consists of an encoder and a decoder with identical structures to \( f_t \) and \( h_t \), respectively, for task \( t \). We also compare with the widely-used Hard Parameter

1. The last assumption in Assumption 1 is not essential but it can help give simpler theoretical results as verified by the proofs in the appendix.
Sharing (HPS) model, which consists of a shared encoder and task-specific decoders with the network structures identical to $f_S$ and $\{h_t\}_{t=1}^m$, respectively. Then the empirical loss of the STL model on task $t$ is formulated as $L_{t}^{\text{STL}} = \sum_{i=1}^{n} \mathcal{L}(y_i, h_t^{\text{STL}}(f_t^{\text{STL}}(x_i)))$ and the expected loss of the STL model on task $t$ is computed as $E_{t}^{\text{STL}} = \mathbb{E}_{(x,y) \sim \mathcal{D}_{t}}[\mathcal{L}(y, h_t^{\text{STL}}(f_t^{\text{STL}}(x)))]$, where $f_t^{\text{STL}}$ and $h_t^{\text{STL}}$ have identical network structures to $f_t$ and $h_t$ in DSMLT, respectively. The average empirical loss of the STL model is computed as $L^{\text{STL}} = \frac{1}{m} \sum_{t=1}^{m} L_{t}^{\text{STL}}$, and the average expected loss of the STL model is as $E^{\text{STL}} = \frac{1}{m} \sum_{t=1}^{m} E_{t}^{\text{STL}}$. Similarly, the average empirical loss of the HPS model is formulated as $L^{\text{HPS}} = \frac{1}{mn} \sum_{i=1}^{n} \sum_{t=1}^{m} \mathcal{L}(y_i, h_t^{\text{HPS}}(f_S^{\text{HPS}}(x_i)))$, where $h_t^{\text{HPS}}$ and $f_S^{\text{HPS}}$ have identical network structures to $h_t$ and $f_S$ in DSMLT, respectively.

The expected loss of DSMLT on task $t$ is formulated as $E_t = \mathbb{E}_{(x,y) \sim \mathcal{D}_t}[\mathcal{L}(y, h_t(g_t(f_S(x), f_t(x)))].$ Then the average expected loss of the DSMLT model is computed as $E = \frac{1}{m} \sum_{t=1}^{m} E_t$.

### 5.2 Analysis on DSMLT-IL

In DSMLT-IL, since $\alpha_t$ is set to zero for each task in the first stage, the objective function of the first stage is equivalent to the following problem as

$$
\min_{\Theta_S} \frac{1}{mn} \sum_{t=1}^{m} \sum_{i=1}^{n} \mathcal{L}(y_i, h_t(g_t^{0}(\emptyset, f_t(x_i))))
$$

where $g_t^{0}$ denotes the gate of task $t$ with $\alpha_t$ as 0 and $\emptyset$ denotes a null network. Thus the first stage is to train the STL model with the empirical loss $L_{t}^{\text{STL}}$ for task $t$. As $g_t$ is a learnable gate, after sufficient training in the second stage, the empirical loss of the DSMLT-IL model on each task is no larger than that of the first stage. Based on this observation, we have the following theorem.

**Theorem 1.** Let $L^*$ be the optimal value of problem (3) and $L_{t}^{\ast}$ the corresponding empirical loss for task $t$. Then we have $L_{t}^{\ast} \leq L_{t}^{\text{STL}}$ for all $1 \leq t \leq m$.

**Theorem 2.** Suppose Assumption 2 is satisfied. Let $L^*$ be the optimal value of problem (3) and the corresponding solution is denoted by $f_S, \{f_t\}, \{h_t\}, \{\hat{g}_t\}$. Let $\hat{E}_t = E_t(f_S, f_t, h_t, \hat{g}_t)$. Then for $(X, Y) \sim \prod_{t=1}^{m} (\mu_t^n)$, with probability at least $1 - \delta$, we have

$$
\hat{E}_t + \epsilon_t \leq E_t^{\text{STL}} + \rho_{n,t},
$$

where $\epsilon_t$ is formulated as $\epsilon_t = L_{t}^{\text{STL}} - L_{t}^{\ast}$, $L_{t}^{\text{STL}}$ is defined as $L_{t}^{\text{STL}} = \mathbb{E}_{(x,y) \sim \mathcal{D}_t}[\mathcal{L}(y, h_t^{\text{STL}}(f_t^{\text{STL}}(x)))]$, $\rho_{n,t}$ is defined as $\rho_{n,t} = C_1 G(\mathcal{F}(X)) + C_2 Q + \sqrt{\frac{\ln \frac{1}{\delta}}{n}}$, $G(\cdot)$ denotes the Gaussian average, and $Q$ is defined as $Q = \sup_{\gamma \in \mathcal{G}} \sup_{\gamma \in \mathcal{H}} \sup_{h_t \in \mathcal{H}} \mathbb{E}_{(x,y) \sim \mathcal{D}_t}[\gamma(h_t(x)) - h_t(x)]$ with $\gamma$ as a vector of independent standard normal variables.

**Remark 5.** For many classes of interest, the Gaussian average $G(\mathcal{F}(X))$ is $O(\sqrt{n})$ according to (36). For reasonable classes $\mathcal{H}$, one can find a bound on $Q$, which is independent of $n$ (37). Therefore, $\rho_{n,t} = O(1/\sqrt{n})$ for $1 \leq t \leq m$ and hence $\rho_{n,t}$ satisfies $\lim_{n \to \infty} \rho_{n,t} = 0$. Moreover, according to Theorem 4, we have $\epsilon_t \geq 0$. Thus, Theorem 2 proves that the proposed DSMLT-IL model can achieve probably individual safe multi-task learning in Definition 3.

### 5.3 Analysis on DSMTL-JL

In this section, we analyze the safety and excess risk bound of the DSMLT-JL model.

For the safety of the DSMLT-JL model, we have the following theorems.

**Theorem 3.** Let $L^*$ be the optimal value of problem (1). Then we have $L^*$ is no higher than the minimum of $L_{t}^{\text{STL}}$ and $L_{t}^{\text{HPS}}$, i.e.,

$$
L^* \leq \min(L_{t}^{\text{STL}}, L_{t}^{\text{HPS}}),
$$

**Remark 6.** Theorem 3 shows that the DSMLT-JL model can achieve empirically average safe multi-task learning in Definition 3. Moreover, it also implies that it can achieve a lower average empirical loss compared with the corresponding HPS model. To see that, the HPS model for task $t$ can be represented as $h_t(g_t^{1}(f_S(x), \emptyset))$, where $g_t^{1}$ denotes the gate of task $t$ with $\alpha_t$ as 1. As $g_t^{1}$ is a feasible solution for the DSMLT-JL model, it is easy to see that the empirical loss of the DSMLT-JL model after sufficient training is lower than that of the HPS model, which could be one reason why the DSMLT-JL model outperforms the HPS model as shown in experiments.

**Theorem 4.** Suppose Assumption 1 is satisfied. Let $L^*$ be the optimal value of problem (1) and the corresponding solution is denoted by $f_S, \{f_t\}, \{h_t\}, \{\hat{g}_t\}$. Let $\hat{E} = E(f_S, \{f_t\}, \{h_t\}, \{\hat{g}_t\})$. Then for $(X, Y) \sim \prod_{t=1}^{m} (\mu_t^n)$, with probability at least $1 - \delta$, we have

$$
\hat{E} + \epsilon \leq E_t^{\text{STL}} + \rho_{n,t},
$$

where $\rho_{n,t} = C_3 G(\mathcal{F}(X)) + C_4 Q + \sqrt{\frac{18 \ln 2}{n}}$, $\mathcal{F}(X) = \{\{f(x_i)\} : f \in \mathcal{F}\}$, $C_1$ and $C_2$ are two constants, $\epsilon$ is formulated as $\epsilon = L_{t}^{\text{STL}} - L^*$, and $Q$ is defined in Theorem 2.

**Remark 7.** According to (36), the Gaussian average $G(\mathcal{F}(X))$ is $O(\sqrt{n}/\sqrt{m})$ for many classes of interest. Therefore, $\rho_{n,t} = O(1/\sqrt{n})$ and it satisfies $\lim_{n \to \infty} \rho_{n,t} = 0$. According to Theorem 3, $\epsilon \geq 0$. Thus, Theorem 4 implies that the proposed DSMLT-JL model can achieve probably average safe multi-task learning in Definition 4.

To analyze the excess risk bound for the DSMLT-JL model, we define the minimal expected risk as

$$
\mathcal{E}^* = \min_{f_S, \{f_t\}, \{h_t\} \in \mathcal{M}} \mathcal{E}.
$$

Then we have the following result.

**Theorem 5.** Suppose Assumption 4 is satisfied. The solution of the optimization problem in Eq. (1) is denoted by $f_S, \{f_t\}, \{h_t\}$,
both the public and private encoders have the same number of parameters. Without loss of generality, we assume that private encoders as a whole, we divide them into modules, consisting of three modules (i.e., $P = 3$). In the retraining process, only the branch with the largest $\beta_t$ is preserved and the rest are removed. Since $\beta_t$ is a binary variable, this discrete nature makes the stochastic gradient descent methods incapable of learning them. Here we relax $\beta_t$ to be continuous and define them as the probability of branching at each branch position. Specifically, the output of the combined encoder for task $t$ is defined as

$$
\alpha_t(x, \beta_t) = \sum_{m=1}^{P} \beta_t^m f_t(x, \beta_t^m),
$$

where $\beta_t^m$ is in the $(P - 1)$-dimensional simplex set denoted by $S_P$, satisfying that $\beta_t^m \geq 0$ and $\sum_{m=1}^{P} \beta_t^m = 1$. Here $\alpha_t(x, \beta_t)$ is a convex combination of outputs of all possible branching architectures weighted by probabilities based on $\beta_t$. Then $\alpha_t(x, \alpha_t)$ is fed into the decoder $h_t$ to generate the prediction and hence the empirical loss for task $t$ is formulated as

$$
L_t(\Theta_t, \beta_t) = \frac{1}{n} \sum_{i=1}^{n} L_t(y_i^t, h_t(\alpha_t(x_i^t, \beta_t))).
$$

where $\Theta_t$ includes all the parameters in $f_s, f_t$, and $h_t$. Then the weighted empirical loss over $m$ tasks is formulated as

$$
L(\Theta, \beta, w) = \sum_{t=1}^{m} w_t L_t(\Theta_t, \beta_t),
$$

where $\Theta = \{\Theta_t\}_{t=1}^{m}$, $w = \{w_1, \ldots, w_m\}$ is in $S_m$ satisfying $w_t \geq 0$ and $\sum_{t=1}^{m} w_t = 1$, and $\beta = (\beta_1, \ldots, \beta_m)$. $w_t$ size. The search space for the architecture in the DSMTL-AL method consists of $m$ architecture parameters $\{\beta_t\}_{t=1}^{m}$ to decide branch positions for $m$ tasks, where $\beta_t = (\beta_t^1, \ldots, \beta_t^m)$. As a binary parameter, $\beta_t^m \in \{0, 1\}$ indicates whether task $t$ branches at the branch position $p$ from $f_s$ to $f_t$. Specifically, when $\beta_t^m$ equals 1, there will be a branch to feed the output of the $(p - 1)$-th module in $f_s$ to the $p$-th module in $f_t$ to form a combined encoder for task $t$. Moreover, the sum of entries in $\beta_t$ should be 1 for any $t$, indicating that there is only one branch position for each task. In this sense, only part of the private/public modules in all the encoders will be used for each task. Hence the model size is smaller than the entire supernet, which is used in the DSMTL-IL and DSMTL-JL models.

The search space in the DSMTL-AL method includes both STL and HPS architectures as two extremes. When all the tasks are unrelated to each other, the architecture in the DSMTL-AL method could become the STL architecture by choosing $f_s$ for each task (i.e., $\beta_t^1 = 1$ for $t = 1, \ldots, m$). For highly related or even identical tasks, the architecture of the DSMTL-AL method could become the HPS architecture by choosing $f_s$ only (i.e., $\beta_t^m = 1$ for $t = 1, \ldots, m$).

The DSMTL-AL method is to find the best branching architecture in the search space for all the $m$ tasks by learning architecture parameters $\{\beta_t\}_{t=1}^{m}$. If task $t$ branches at the branch position $p$ of $f_s$ to connect to the corresponding next module in $f_t$, $\beta_t^p$ is set to 1 and all the $\beta_t^i$’s ($i \neq p$) are set to 0. In this case, the output of the combined encoder for task $t$ consisting of the first $(p - 1)$ public modules in $f_s$ and the last $(P - p)$ private modules in $f_t$ is $f_t(f_s(x, p - 1), p, P - 1)$, where $f_s(\cdot, p)$ denotes the output of the $p$-th module in $f_s$ and $f_t(\cdot, p, q)$ denotes the output of the $q$-th module in $f_t$ starting from the $p$-th module. Here $f_s(0, 0)$ is defined to be $x$, corresponding to the input to the first module, and $f_t(x, P, P - 1)$ is defined as $x$. Since $\{\beta_t\}$ are binary variables, this discrete nature makes stochastic gradient descent methods incapable of learning them. Here we relax $\{\beta_t\}$ to be continuous and define them as the probability of branching at each branch position. Specifically, the output of the combined encoder for task $t$ is defined as

$$
\alpha_t(x, \beta_t) = \sum_{m=1}^{P} \beta_t^m f_t(x, P, p, P - 1),
$$

where $\beta_t$ is in the $(P - 1)$-dimensional simplex set denoted by $S_P$, satisfying that $\beta_t^m \geq 0$ and $\sum_{m=1}^{P} \beta_t^m = 1$. Here $\alpha_t(x, \beta_t)$ is a convex combination of outputs of all possible branching architectures weighted by probabilities based on $\beta_t$. Then $\alpha_t(x, \alpha_t)$ is fed into the decoder $h_t$ to generate the prediction and hence the empirical loss for task $t$ is formulated as

$$
L_t(\Theta_t, \beta_t) = \frac{1}{n} \sum_{i=1}^{n} L_t(y_i^t, h_t(\alpha_t(x_i^t, \beta_t))),
$$

where $\Theta_t$ includes all the parameters in $f_s, f_t$, and $h_t$. Then the weighted empirical loss over $m$ tasks is formulated as

$$
L(\Theta, \beta, w) = \sum_{t=1}^{m} w_t L_t(\Theta_t, \beta_t),
$$

where $\Theta = \{\Theta_t\}_{t=1}^{m}$, $w = \{w_1, \ldots, w_m\}$ is in $S_m$ satisfying $w_t \geq 0$ and $\sum_{t=1}^{m} w_t = 1$, and $\beta = (\beta_1, \ldots, \beta_m)$. $w_t$
specifies the weighting among all the tasks. Setting them to \( \frac{1}{n} \) as in the DSMTL-IL and DSMTL-JL models may lead to suboptimal performance and hence we aim to learn them directly.

Here \( \Theta \) is viewed as model parameters, while \( \beta \) and \( w \) are hyperparameters. To learn all of them, we adopt a bi-level formulation as

\[
\min_{\beta \in \mathcal{S}_\beta, w \in \mathcal{S}_w} L_{\text{val}}(\hat{\Theta}, \beta, w)
\]

subject to

\[
\hat{\Theta} = \arg \min_{\Theta} L_{\text{tr}}(\Theta, \beta, w),
\]

where the entire training dataset is divided into a training set and a validation set, \( L_{\text{tr}}(\cdot, \cdot, \cdot) \) denotes the weighted empirical loss defined in Eq. (7) on the training set, and \( L_{\text{val}}(\cdot, \cdot, \cdot) \) denotes the weighted empirical loss defined in Eq. (7) on the validation set. Here the constraints on \( \beta \) and \( w \) can be alleviated via the reparameterization based on the softmax function. We adopt the gradient-based hyperparameter optimization algorithm in [11], [30] to solve problem (8) with the first-order approximation. After solving the problem (8), we can learn architecture for task \( t \) by determining the branch position as

\[
\beta^*_t = \arg \max_p (\{\beta^*_t\}_p=1).
\]

With the learned architecture, we can use the entire training dataset to retrain the model parameters \( \Theta \). Inspired by the gating mechanism in the DSMTL-IL and DSMTL-JL models, the final encoder for task \( t \) consists of the shared encoder and the combined encoder determined by \( \beta^*_t \). One reason for that is that the shared encoder \( f_S \) could be fully used to improve the performance with little or even no increase in the model size, which is due to that all the modules in \( f_S \) will usually be chosen by at least one task during the architecture learning process. Formally, the final encoder for task \( t \) is formulated as

\[
\hat{g}_t(x, \beta^*_t) = \alpha_t f_S(x) + (1 - \alpha_t) f_t(f_S(x, \beta^*_t - 1), \beta^*_t, P - 1),
\]

where with abuse of notations, \( \alpha_t \in \mathcal{M} = [0, 1] \) is a learnable parameter to measure the weight of \( f_S(x) \) and acts similarly to \( \alpha_t \) in the DSMTL-IL and DSMTL-JL models. Mathematically, the objective function of the retraining process is formulated as

\[
\min_{\Theta, \alpha \in \mathcal{M}, \gamma \in \mathcal{M}} \sum_{t=1}^m \sum_{i=1}^n w_t \gamma \mathcal{L}_t(y_i^t, h_t(\hat{g}_t(x_i^t, \beta^*_t))),
\]

where \( \mathcal{L}_t \) denotes all the model parameters, \( \alpha = (\alpha_1, \ldots, \alpha_m) \), and \( w_t \) is the loss weight learned from problem (9) for task \( t \). After the retraining process, we can use the learned model parameters to make prediction for each task.

Though the DSMTL-AL model cannot be theoretically proved to achieve some version of safe multi-task learning, as shown in the next section, empirically it not only achieves good performance but also learns compact architectures.

### 7 Experiments

In this section, we evaluate the proposed models.

#### 7.1 Datasets and Evaluation Metrics

Experiments are conducted on four MTL CV datasets, including CityScapes [8], NYUv2 [44], PASCAL-Context [40], and Taskonomy [57].

The CityScapes dataset consists of high resolution outside street-view images. By following [31], we evaluate the performance on the 7-class semantic segmentation and depth estimation tasks. The NYUv2 dataset consists of RGB-D indoor scene images from three learning tasks: 13-class semantic segmentation, depth estimation, and surface normal prediction. The PASCAL-Context dataset is an annotation extension of the PASCAL VOC 2010 challenge with four learning tasks: 21-class semantic segmentation, 7-class human parts segmentation, saliency estimation, and surface normal estimation, where the last two tasks are generated by [33]. The Taskonomy dataset contains indoor images. By following [45], we sample five learning tasks, including 17-class semantic segmentation, depth estimation, keypoint detection, edge detection, and surface normal prediction.

The semantic segmentation task on the PASCAL-Context dataset is evaluated by the mean Intersection over Union (mIoU) by following [35]. On the other three CV datasets, this task is additionally evaluated in terms of the Pixel Error (abbreviated as ‘Pix Err’) by following [45]. For the depth estimation task, the absolute error (abbreviated as ‘Abs Err’) and relative error (abbreviated as ‘Rel Err’) are used as the evaluation metrics. For the surface normal prediction task, the mean and median angle distances between the prediction and ground truth of all pixels are used as measures. For this task, the percentage of pixels, whose prediction is within the angles of 11.25°, 22.5°, and 30° to the ground truth, is used as another measure. For the keypoint detection and edge detection tasks, the absolute error (abbreviated as ‘Abs Err’) is used as the evaluation metric. For the human parts segmentation task, the mIoU is used as the measure. For the saliency estimation task, the mIoU and max F-measure (maxF) are adopted as the evaluation metrics.

As introduced above, for each task, we use one or more evaluation metrics to thoroughly evaluate the performance. To better show the comparison between each method and STL, we compute the relative performance of each method over STL in terms of the \( j \)-th evaluation metric on task \( t \) as \( \Delta_{t,j} = (1 - \psi_{t,j}) (M_{t,j} - \text{STL}_{t,j}) \), where for a method \( M \), \( M_{t,j} \) denotes its performance in terms of the \( j \)-th evaluation metric for task \( t \), STL\(_{t,j}\) is defined similarly, \( \psi_{t,j} \) equals 1 if a lower value represents better performance in terms of the \( j \)-th metric in task \( t \) and 0 otherwise. So positive relative performance indicates better performance than STL. The overall relative improvement of a method \( M \) over STL is defined as \( \Delta_t = \frac{1}{m} \sum_{t=1}^m \frac{1}{m_j} \sum_{j=1}^n \Delta_{t,j} \), where \( m_t \) denotes the number of evaluation metrics in task \( t \).

To empirically measure the safeness of each model, we define the safeness coefficient \( \eta \) for a model as the proportion of tasks on which this model empirically performs no worse than the STL model. Formally, \( \eta \) is formulated as

\[
\eta = \frac{1}{m} \sum_{t=1}^m \frac{1}{m_t} \sum_{j=1}^n \delta(\Delta_{t,j}) \times 100,
\]

where \( \delta(x) \) is the delta function that outputs 0 when \( x < 0 \) and otherwise 1. Obviously, \( \eta \), whose maximum is 100, is expected to be as large as possible.
7.2 Experimental Setup

The baseline methods in comparison include the Single-Task Learning (STL) that trains each task separately, popular MTL architectures including the HPS model that adopts the multi-head hard sharing architecture, Cross-stitch [38], MTAN [31], NDDR-CNN [13], RotoGrad [19], and AFA [9], and popular architecture learning methods for MTL such as MaxRoam [41], TSN [47], MTL-NAS [12], BMTAS [3], and LTB [16]. For fair comparison, we use the same backbone for all the models. Similar to [31], we use the Deeplab-ResNet [6] with atrous convolutions as encoders and the ASPP architecture [6] as decoders. We adopt the ResNet-50 pretrained on ImageNet for the CityScapes and NYUv2 datasets to implement the the Deeplab-ResNet, and use the pretrained ResNet-18 on the larger PASCAL-Context and Taskonomy datasets for training efficiency. We use the cross-entropy loss for the semantic segmentation, human parts segmentation and saliency estimation tasks, the cosine similarity loss for the surface normal prediction task, and the L1 loss for other tasks. For the DSMTL-AL method, a module is defined as a layer in the Deeplab-ResNet model and the branch position is set before and after each layer. Therefore, we have five layers (including conv1) and have six branch positions. For optimization, we use the Adam method [21] with the learning rate as $10^{-4}$. All the experiments are conducted on Tesla V100 GPUs.

7.3 Experimental Results

Tables [1] show the performance of all the models in comparison on different datasets. On the CityScapes dataset, the proposed DSMTL-IL, DSMTL-JL, DSMTL-AL and some baseline methods (i.e., Cross-stitch, MTAN, RotoGrad, BMTAS, TSN, and NDDR-CNN) perform no worse than the STL model on each metric and hence under such setting, they achieve safe multi-task learning (i.e., $\eta = 100$). In addition, the proposed DSMTL-AL model achieves the best $\Delta_I$, which demonstrates its effectiveness. On the NYUv2 and PASCAL-Context datasets, none of the baselines can achieve safe multi-task learning, while the proposed methods (i.e., DSMTL-IL, DSMTL-JL, and DSMTL-AL) can achieve that, which again shows the effectiveness of the proposed methods.

7.4 Analysis on the Position of Gate

In this section, we study how the position of the gate affects the performance of the proposed models and we use the DSMTL-JL model as an example. As there are $P-1$ possible positions for each gate by excluding the position before the first layer, we try each of them to see which one is the best in terms of the performance. To avoid the exponential complexity, we assume that gate positions of different tasks are the same. Specifically, we put a gate $g_i(\cdot, p)$ after the $p$-
th module of the private encoder in task t and the entire encoder for task f is formulated as
\[ g_t(x, p) = f_t(\alpha^p_t f_s(x, p) + (1 - \alpha^p_t) f_t(x, 1, p), p + 1, P - 1), \]
where as defined in Section 6, \( f_s(\cdot, p) \) denotes the output of the \( p \)-th module in \( f_s \) and \( f_t(\cdot, p, q) \) denotes the output of the \( q \)-th module in \( f_t \) starting from its \( p \)-th module.

According to Figure 6, when changing the gate position from \( p = 2 \) to \( p = 5 \), the performance of the DMTL-JL model becomes better, and \( p = 5 \) gives the best performance, which justifies the choice of the gate position in the DMTL-JL and DMTL-JL models and also inspires the design of the final encoder in the DMTL-AL method as defined in Eq. 9.

### 7.5 Analysis on Learned Task Relevance

We show the learned \( \{\alpha_t\} \) of the proposed methods in Table 5. According to the results, we can see that some \( \alpha_t \)'s are close to 0.5, which implies that in those cases, the public encoder and the private encoder are both important to the corresponding tasks. Thus, only using the public encoder (i.e., HPS) and only using the private encoder (i.e., STL) cannot achieve good performance, while the proposed models can take the advantages of these two methods to achieve better performance in most cases. Moreover, some of the learned \( \alpha_t \)'s have relatively small values (i.e., values smaller than 0.3), which are shown in box. These small
each dataset, at least one task choose to use the entire shared
methods have good performance on those datasets (refer to
datasets, which aligns with the task relationship founded in
consistent with the architecture learned by the DSMTL-AL
values indicate that for the surface normal prediction task
encoder, which corresponds to the choice of the branch point
As the loss scales and converge speed of different tasks
show that loss weights are uneven. For example, the surface
encoder, which corresponds to the choice of the branch point
pared with the original DSMTL methods, the performance
erage of outputs of both public and private encoders. Com-
normal prediction task has a larger loss weight than the
model size.

\[ \alpha \]

\[ \eta \]

\[ \text{SS} \]

\[ \text{SN} \]

\[ \text{STL} \]

\[ \text{Cross-stitch} \]

\[ \text{DSMTL-JL} \]

\[ \text{DSMTL-IL} \]

\[ \text{DSMTL-AL} \]

\[ \text{HPS} \]

\[ \text{Cross-stitch} \]

\[ \text{MTAN} \]

\[ \text{NDDR-CNN} \]

\[ \text{AFA} \]

\[ \text{RotoGrad} \]

\[ \text{MaxRoom} \]

\[ \text{MTL-NAS} \]

\[ \text{BMTAS} \]

\[ \text{TSN} \]

\[ \text{LTT} \]

\[ \text{DE} \]

\[ \text{SNP} \]

\[ \text{DSMTL-IL} \]

\[ \text{DSMTL-JL} \]

\[ \text{DSMTL-AL} \]

\[ \text{CityScapes} \]

\[ \text{NYUv2} \]

\[ \text{PASCAL-Context} \]

\[ \text{Taskonomy} \]

\[ \text{Parm. (M)} \]

\[ \Delta \]

\[ \eta \]

\[ \alpha \]

\[ \omega \]

\[ \text{SS} \]

\[ \text{SN} \]

\[ \text{STL} \]

\[ \text{Cross-stitch} \]

\[ \text{MTAN} \]

\[ \text{NDDR-CNN} \]

\[ \text{AFA} \]

\[ \text{RotoGrad} \]

\[ \text{MaxRoom} \]

\[ \text{MTL-NAS} \]

\[ \text{BMTAS} \]

\[ \text{TSN} \]

\[ \text{LTT} \]

\[ \text{DE} \]

\[ \text{SNP} \]

\[ \text{DSMTL-IL} \]

\[ \text{DSMTL-JL} \]

\[ \text{DSMTL-AL} \]

\[ \text{CityScapes} \]

\[ \text{NYUv2} \]

\[ \text{PASCAL-Context} \]

\[ \text{Taskonomy} \]

\[ \text{Parm. (M)} \]

\[ \Delta \]

\[ \eta \]

\[ \alpha \]

\[ \omega \]

\[ \text{SS} \]

\[ \text{SN} \]

\[ \text{STL} \]

\[ \text{Cross-stitch} \]

\[ \text{MTAN} \]

\[ \text{NDDR-CNN} \]

\[ \text{AFA} \]

\[ \text{RotoGrad} \]

\[ \text{MaxRoom} \]

\[ \text{MTL-NAS} \]

\[ \text{BMTAS} \]

\[ \text{TSN} \]

\[ \text{LTT} \]

\[ \text{DE} \]

\[ \text{SNP} \]

\[ \text{DSMTL-IL} \]

\[ \text{DSMTL-JL} \]

\[ \text{DSMTL-AL} \]

\[ \text{CityScapes} \]

\[ \text{NYUv2} \]

\[ \text{PASCAL-Context} \]

\[ \text{Taskonomy} \]

\[ \text{Parm. (M)} \]

\[ \Delta \]

\[ \eta \]

\[ \alpha \]

\[ \omega \]

\[ \text{SS} \]

\[ \text{SN} \]

\[ \text{STL} \]

\[ \text{Cross-stitch} \]

\[ \text{MTAN} \]

\[ \text{NDDR-CNN} \]

\[ \text{AFA} \]

\[ \text{RotoGrad} \]

\[ \text{MaxRoom} \]

\[ \text{MTL-NAS} \]

\[ \text{BMTAS} \]

\[ \text{TSN} \]

\[ \text{LTT} \]

\[ \text{DE} \]

\[ \text{SNP} \]

\[ \text{DSMTL-IL} \]

\[ \text{DSMTL-JL} \]

\[ \text{DSMTL-AL} \]

\[ \text{CityScapes} \]

\[ \text{NYUv2} \]

\[ \text{PASCAL-Context} \]

\[ \text{Taskonomy} \]

\[ \text{Parm. (M)} \]

\[ \Delta \]

\[ \eta \]

\[ \alpha \]

\[ \omega \]

\[ \text{SS} \]

\[ \text{SN} \]

\[ \text{STL} \]

\[ \text{Cross-stitch} \]

\[ \text{MTAN} \]

\[ \text{NDDR-CNN} \]

\[ \text{AFA} \]

\[ \text{RotoGrad} \]

\[ \text{MaxRoom} \]

\[ \text{MTL-NAS} \]

\[ \text{BMTAS} \]

\[ \text{TSN} \]

\[ \text{LTT} \]

\[ \text{DE} \]

\[ \text{SNP} \]

\[ \text{DSMTL-IL} \]

\[ \text{DSMTL-JL} \]

\[ \text{DSMTL-AL} \]

\[ \text{CityScapes} \]

\[ \text{NYUv2} \]

\[ \text{PASCAL-Context} \]

\[ \text{Taskonomy} \]

\[ \text{Parm. (M)} \]

\[ \Delta \]

\[ \eta \]

\[ \alpha \]

\[ \omega \]

\[ \text{SS} \]

\[ \text{SN} \]

\[ \text{STL} \]

\[ \text{Cross-stitch} \]

\[ \text{MTAN} \]

\[ \text{NDDR-CNN} \]

\[ \text{AFA} \]

\[ \text{RotoGrad} \]

\[ \text{MaxRoom} \]

\[ \text{MTL-NAS} \]

\[ \text{BMTAS} \]

\[ \text{TSN} \]

\[ \text{LTT} \]

\[ \text{DE} \]

\[ \text{SNP} \]

\[ \text{DSMTL-IL} \]

\[ \text{DSMTL-JL} \]

\[ \text{DSMTL-AL} \]

\[ \text{CityScapes} \]

\[ \text{NYUv2} \]

\[ \text{PASCAL-Context} \]

\[ \text{Taskonomy} \]

\[ \text{Parm. (M)} \]

\[ \Delta \]

\[ \eta \]

\[ \alpha \]

\[ \omega \]

\[ \text{SS} \]

\[ \text{SN} \]

\[ \text{STL} \]

\[ \text{Cross-stitch} \]

\[ \text{MTAN} \]

\[ \text{NDDR-CNN} \]

\[ \text{AFA} \]

\[ \text{RotoGrad} \]

\[ \text{MaxRoom} \]

\[ \text{MTL-NAS} \]

\[ \text{BMTAS} \]
decreases in both DSMTL-IL and DSMTL-JL cases, which indicates that learnable gates are a key ingredient for the DSMTL-IL and DSMTL-JL models to achieve the safeness. The safeness coefficient still keeps as 100 in the variant of the DSMTL-AL method, which implies that the DSMTL-AL method can learn a reliable architecture to achieve the safeness.

For "w/o learning task weighting", we replace the learned task weights in the DSMTL-AL method with identical loss weights during the retraining process. This variant without the learnable task weighting has inferior performance to the DSMTL-AL method, which indicates that learning task weighting in the DSMTL-AL method cannot only reduce tedious costs to tune loss weights but also improve the performance of the DSMTL-AL method.

7.7 Combination and Comparison with Loss Weighting Strategies

The loss weighting scheme adopted in the DSMTL-IL and DSMTL-JL methods is the commonly used Equally Weighting (EW) strategy (i.e., all the loss weights are equal to \( \frac{1}{m} \) in problems (3) and (4)). The loss weighting scheme adopted in the DSMTL-AL method is a Learnable Weighting (LW) strategy as in Eq. (7). In this section, we show that the proposed DSMTL models could be combined with some loss weighting methods in MTL, including Uncertainty Weights (UW) [20], Dynamic Weight Average (DWA) [31], Geometric Loss Strategy (GLS) [7], PCGrad [56], and CAGrad [29].

According to experimental results shown in Table 7, the combination of the DSMTL-IL method and other methods than the EW strategy has inferior performance and a lower safeness coefficient \( \eta \), which verifies the usefulness of the EW strategy adopted in the DSMTL-IL method. Differently, the performance of the DSMTL-JL method could be improved in terms of \( \Delta_I \) when combining with some loss weighting strategies (i.e., GLS and CAGrad), and all the combinations have the largest safeness coefficients. One reason for the aforementioned difference between the DSMTL-IL and DSMTL-JL methods is that the DSMTL-IL method fixes all parameters of the private encoders and decoders in the second stage of the training process, making the corresponding parameters not fully updated and therefore leading to the inferior performance. For the DSMTL-AL method, replacing LW with PCGrad can further improve the performance while other loss weighting strategies degrade the performance.

In terms of computational cost or the training speedup over the STL, the PCGrad and CAGrad methods have large computational overhead since they need to project high-dimensional gradients of different tasks on each training step. The UW, DWA, and GLS methods have relatively small computational overhead but with limited performance improvement. The LW strategy in DSMTL-AL has no computational overhead during the retraining process and have competitive performance compared with various loss weighting strategies, which demonstrate the effectiveness of the LW strategy.

8 Conclusion

In this paper, we formally define the problem of safe multi-task learning, and propose a simple and effective DSMTL method that can learn to combine the shared and task-specific representations. We theoretically analyze the proposed models and prove that the proposed DSMTL-IL and DSMTL-JL methods are guaranteed to achieve some versions of safe multi-task learning. To solve the scalability issue of the proposed DSMTL-IL and DSMTL-JL methods, we further propose the DSMTL-AL method to learn a compact architecture via techniques in neural architecture search. Extensive experiments demonstrate the effectiveness of the proposed methods. In the future work, we are interested in generalizing the DSMTL methods to other learning problems.

Acknowledgements

This work is supported by NSFC key grant under grant no. 62136005, NSFC general grant under grant no. 62076118, and Shenzhen fundamental research program JCYJ2021032410500003.

References

[1] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. JMLR, 2002.
[2] Felix JS Bragman, Ryutaro Tanno, Sebastien Ourselin, Daniel C. Alexander, and Jorge Cardoso. Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels. In ICCV, 2019.
[3] David Bruggemann, Menelaos Kanakis, Stamatios Georgoulis, and Luc Van Gool. Automated search for resource-efficient branched multi-task networks. In BMVC, 2020.
[4] Jiajiong Cao, Yingming Li, and Zhongfei Zhang. Partially shared multi-task convolutional neural network with local constraint for face attribute learning. In CVPR, 2018.
[5] R. Caruana. Multitask learning. Machine Learning, 1997.

Table 7

| Architecture | Methods | Train Speedup ↑ | \( \Delta_I \) ↑ | \( \eta \) ↑ |
|--------------|---------|----------------|----------------|----------|
| STL          | -       | 1.0x           | 0              | 100      |
| HPS          | EW      | 1.69x          | +0.0001        | 67       |
|              | UW      | 1.66x          | -0.0040        | 67       |
|              | DWA     | 1.68x          | +0.0029        | 67       |
|              | GLS     | 1.68x          | +0.0158        | 67       |
|              | PCGrad  | 0.78x          | +0.0037        | 67       |
|              | CAGrad  | 0.61x          | +0.0023        | 67       |
| DSMTL-IL     | EW      | 1.18x          | +0.0067        | 100      |
|              | UW      | 1.15x          | +0.0035        | 53       |
|              | DWA     | 1.17x          | +0.0038        | 60       |
|              | GLS     | 1.16x          | +0.0041        | 67       |
|              | PCGrad  | 0.43x          | -0.0397        | 0        |
|              | CAGrad  | 0.36x          | -0.0399        | 0        |
| DSMTL-JL     | EW      | 0.89x          | +0.0221        | 100      |
|              | UW      | 0.83x          | +0.0014        | 100      |
|              | DWA     | 0.84x          | +0.0191        | 100      |
|              | GLS     | 0.88x          | +0.0269        | 100      |
|              | PCGrad  | 0.34x          | +0.0164        | 100      |
|              | CAGrad  | 0.30x          | +0.0257        | 100      |
| DSMTL-AL     | LW      | 1.18x          | +0.0281        | 100      |
|              | EW      | 1.18x          | +0.0166        | 100      |
|              | UW      | 1.14x          | +0.0163        | 100      |
|              | DWA     | 1.16x          | +0.0204        | 93       |
|              | GLS     | 1.17x          | +0.0203        | 80       |
|              | PCGrad  | 0.42x          | +0.0299        | 100      |
|              | CAGrad  | 0.38x          | +0.0243        | 100      |
[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE TPAMI, 2017.

[7] Sumanth Chennupati, Ganesh Sistu, Senthil Yogamani, and Samir A ROWAD. Multinets+: Multi-stream feature aggregation and geometric loss strategy for multi-task learning. In CVPR Workshops, 2019.

[8] Marius Cords, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The citiescape dataset for semantic urban scene understanding. In CVPR, 2016.

[9] Chaoran Cui, Zhen Shen, Jin Huang, Meng Chen, Mingxiang Lu, Meng Wang, and Yilong Yin. Adaptive feature aggregation in deep multi-task convolutional neural networks. IEEE TCSVT, 2021.

[10] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi-task learning. In KDD, 2004.

[11] Luca Franceschi, Paolo Frasconi, Saverio Saltz, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In ICML, 2018.

[12] Yuan Gao, Haoping Bai, Zegun Jie, Jiayi Ma, Kui Jia, and Wei Liu. MTVL-NAS: task-agnostic neural architecture search towards general-purpose multi-task learning. In CVPR, 2020.

[13] Yuan Gao, Jiayi Ma, Mingbo Zhao, Wei Liu, and Alan L Yuille. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In CVPR, 2019.

[14] Lan-Zhe Guo, Zhe-Yu Zhang, Yuan Jiang, Yu-Feng Li, and Zhi-Hua Zhou. Safe deep semi-supervised learning for unseen-class unlabeled data. In ICML, 2020.

[15] Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. In ICML, 2020.

[16] Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. In ICML, 2020.

[17] Pengxin Gao, Chang Deng, Linjie Xu, Xiaonian Huang, and Yu Zhang. Deep multi-task augmented feature learning via hierarchical graph neural network. In ECML PKDD, 2021.

[18] Hu Han, Anil K Jain, Fang Wang, Shiguang Shan, and Xilin Chen. Heterogeneous face attribute estimation: A deep multi-task learning approach. IEEE TPAMI, 2017.

[19] Adrián Javaloy and Isabel Valera. Rotograd: Gradient homogeneity in multitask learning. In ICLR, 2021.

[20] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning with attention. In CVPR, 2019.

[21] Andreas Maurer. A chain rule for the expected suprema of general-purpose multi-task learning. In ICML, 2020.

[22] Yu Zhang and Dit-Yan Yeung. A convex formulation for learning multi-gate mixture-of-experts. In KDD, 2017.

[23] Kai-Wei Chang, Yehua Wang, and Dit-Yan Yeung. Multi-task warped gaussian process for personalized age estimation. In CVPR, 2010.
APPENDIX A

PROOFS

In this section, we provide proofs for all the theorems.

A.1 Generalization Bound for Problem 4

To help analyze the generalization bound of the DSMTL method, we first introduce a useful theorem in terms of Gaussian averages [1], [37].

**Theorem 6.** Let \( \mathcal{G} \) be a class of functions \( \vartheta : \mathcal{X} \to [0, 1] \), and \( \mu_1, \ldots, \mu_m \) be the probability measure on \( \mathcal{X} \) with \( \bar{X} = (X_1, \ldots, X_m) \sim \prod_{i=1}^m (\mu_i)^n \), where \( X_i = (x_{i1}, \ldots, x_{in}) \). Let \( \gamma \) be a vector of independent standard normal variables and \( Z \) be the random set \( \{(\vartheta_i(x_i)): \vartheta \in \mathcal{G}\} \) where \( \vartheta_i \) are functions chosen from hypothesis class \( \mathcal{G} \). Then for all \( \vartheta \in \mathcal{G} \), with probability at least \( 1 - \delta \), we have

\[
\frac{1}{m} \sum_{i} \left( \mathbb{E}_{x \sim \mu_i} [\vartheta_i(x)] - \frac{1}{n} \sum_{i} \vartheta_i(x_i) \right) \leq \sqrt{2\pi} G(Z) \frac{1}{m} + \sqrt{\frac{9 \ln \frac{2}{\delta}}{2mn}},
\]

where \( G(Z) = \mathbb{E}[\sup_{z \in Z} \langle \gamma, z \rangle] \) is the Gaussian average of the random set \( Z \).

Based on Theorem 6, we first establish the following uniform bound for problem 4.

**Theorem 7.** Suppose Assumption 4 is satisfied. Then for \( (X, Y) \sim \prod_{i=1}^n (\mu_i)^p \), with probability at least \( 1 - \delta \), we have

\[
\mathbb{E} - \frac{1}{mn} \sum_{i} \mathcal{L}_i(y_i', h_t(g_t(f_s(x_i'), f_t(x_i')))) \leq \frac{C_1G(F(X))}{\sqrt{n}} + \frac{C_2Q}{\sqrt{n}} + \sqrt{\frac{9 \ln \frac{2}{\delta}}{2mn}},
\]

where \( C_1, C_2 \) are two constants, the quantity \( Q \) is defined as

\[
Q = \sup_{z, z' \in \mathbb{R}^m} \left| z - z' \right| \mathbb{E} \sup_{h_t \in H} \sum_{i=1}^n \gamma_i(h_t(z_i) - h_t(z'_i)),
\]

and \( \gamma \) is a vector of independent standard normal variables.

**Proof.** According to Theorem 6, for \( h_t \in H, f_s, f_t \in F \), and \( \alpha_t \in M \), with probability at least \( 1 - \delta \), we have

\[
\mathbb{E} - \frac{1}{mn} \sum_{i} \mathcal{L}_i(y_i', h_t(g_t(f_s(x_i'), f_t(x_i')))) \leq \sqrt{2\pi} G(S) \frac{1}{mn} + \sqrt{\frac{9 \ln \frac{2}{\delta}}{2mn}},
\]

where \( S = \{(\mathcal{L}_i(y_i', h_t(g_t(f_s(x_i'), f_t(x_i'))))): h_t \in H, \alpha_t \in M, f_s, f_t \in F\} \).

Note that the input data \( X \in \mathcal{X}^m \) and the encoders \( f_1, \ldots, f_m, f_s, f_t : \mathcal{X} \to \mathbb{R}^q \) are mapping functions chosen from \( F \), the random set \( K(X) \subseteq \mathbb{R}^{mnq} \) is defined as \( K(X) = \{g_t(f_s(x_i'), f_t(x_i')): \alpha_t \in M, f_s, f_t \in F\} \). We define a class of functions \( \mathcal{H} = \{z \in \mathbb{R}^{mnq} \mapsto h_t(z_i): h_t \in H\} \). Therefore, we have \( S'(X) = H'(K(X)) \).

By using Theorem 2 in [36], we obtain

\[
G(S') \leq c_1L(H')(G(K(X)) + c_2D(K(X))Q(H')) + \min_{z \in K(X)} G(H'(z)),
\]

where \( c_1 \) and \( c_2 \) are two constants, \( L(H') \) denotes the Lipschitz constant of the functions in \( H' \), \( D(K(X)) = 2\sup_{x \in K} ||\varphi(x)|| \) denotes the Euclidean diameter of the set \( F(X) \), and

\[
Q(H') = \sup_{z \neq z' \in \mathbb{R}^{mnq}} \left| z - z' \right| \mathbb{E} \sup_{h_t \in H'} \langle \gamma, z - \gamma, z' \rangle.
\]

Let \( z, z' \in \mathbb{R}^{mnq} \), where \( z = (z_i'), z' = (z'_i) \) and \( z_i, z'_i \in \mathbb{R}^q \). Then for any functions \( \psi \in H' \), we have

\[
E \sup_{h_t \in H'} \langle \gamma, \varphi(z) - \varphi(z') \rangle
\]

\[
= \sum_{i=1}^m E \sup_{h_t \in H} \sum_{i=1}^n \gamma_i(h_t(z_i) - h_t(z'_i))
\]

\[
\leq \sqrt{m} \left( \sum_{i=1}^m \left( \mathbb{E} \sup_{h_t \in H} \sum_{i=1}^n \gamma_i(h_t(z_i) - h_t(z'_i)) \right) \right)^{1/2}
\]

\[
\leq \sqrt{m} \left( \sum_{i=1}^m Q^2 \sum_{i=1}^n ||z_i - z'_i||^2 \right)^{1/2}
\]

\[
= \sqrt{m}Q \left| z - z' \right|,
\]

where the first inequality is due to the Cauchy-Schwarz inequality and the second inequality is due to the inequality \( \left| z - z' \right| \leq \sup \left| z - z' \right| \) holds for all \( z \neq z' \in \mathbb{R}^{mnq}. \) Therefore, \( Q(H') \leq \sqrt{m}Q. \) Moreover, suppose the functions in hypothesis classes \( H \) are \( M \)-Lipschitz continuous, we have

\[
\left| \varphi(z) - \varphi(z') \right|^2 \leq \sum_{i=1}^m (h_t(z_i) - h_t(z'_i))^2 \leq M^2 \left| z - z' \right|^2
\]

where the inequality is due to the Lipschitz property. So we obtain \( L(H') \leq M. \) Since \( 0 \in F \), we have \( 0 \in K(X). \) Note that \( h(0) = 0 \), and hence \( \min_{z \in F(X)} G(H'(z)) = 0 \) by setting \( z = 0. \) Therefore, we have

\[
G(S) \leq c_1MG(K(X)) + 2c_2\sqrt{m}Q \sup_{\varphi \in K} ||\varphi||. \quad (12)
\]

Recall that the random set \( F(X) \subseteq \mathbb{R}^{mnq} \) is defined as \( F(X) = \{(f_s(x_i')): f_s \in F\} \). The \( j \)th entry of the vectors \( f_s(x_i') \) and \( f_t(x_i') \) by \( f_{s,j}(x_i') \) and \( f_{t,j}(x_i') \), respectively, and let \( \gamma_{j,t,i} \) be the corresponding standard normal variable. Then we have

\[
G(K(X)) = E \left[ \sup_{f_s, f_t, \alpha_t} \sum_{j=1}^q \sum_{i=1}^n \gamma_{j,t,i} g_t(f_s(x_i'), f_t(x_i')) | x_i' \right] \]

\[
= E \left[ \sup_{f_s \in F} \sum_{j=1}^q \sum_{i=1}^n \gamma_{j,t,i} f_{s,j}(x_i') | x_i' \right] = G(F(X)),
\]

where the first and third equality are due to the definition of the Gaussian average, and the second equality holds since \( f_t \) and \( f_s \) are chosen from the same class \( F \) and \( F \) is uniformly bounded.

Since the hypothesis classes \( F \) is uniformly bounded, suppose that \( ||f(x_i')|| \leq R \) for all \( f \in F \) and we have

\[
\sup_{\varphi \in K} ||\varphi|| = \sqrt{mn} \sup_{f_s, f_t, \alpha_t} \max \left| g_t(f_s(x_i'), f_t(x_i')) \right| \leq \sqrt{mn} \sup_{f \in F} \max \left| f(x_i') \right| \leq \sqrt{mn}R,
\]
where the first inequality holds since \( f_t \) and \( f_s \) are chosen from the same class \( \mathcal{F} \), and the second inequality holds since \( \| f(x_i) \| \leq R \). Therefore, we get

\[
\mathcal{E} - \frac{1}{mn} \sum_{t,i} \mathcal{L}_t(y_i^t, h_t(g_t(f_s(x_i), f_t(x_i)))) \\
\leq \frac{c_1 M \sqrt{2\pi G(\mathcal{F}(X))}}{mn} + \frac{2c_2 R \sqrt{2\pi Q}}{\sqrt{m}} + \left( \frac{9 \ln \frac{2}{\delta}}{2mn} \right).
\]

By setting \( C_1 = c_1 M \sqrt{2\pi} \) and \( C_2 = 2c_2 R \sqrt{2\pi} \), we reach the conclusion.

**Remark 8.** According to the McDiarmid’s inequality [39], the upper bound in Theorem 3 also holds for the sum of \( \vartheta_t(x_i) \) minus its expectation. Thus following the same proof as above, we can verify that the following inequality holds under the same assumption in Theorem 6:

\[
\left( \frac{1}{mn} \sum_{t,i} \mathcal{L}_t(y_i^t, h_t(g_t(f_s(x_i), f_t(x_i)))) - \mathcal{E} \right) \leq \frac{C_1 G(\mathcal{F}(X))}{mn} + \frac{C_2 Q}{\sqrt{m}} + \left( \frac{9 \ln \frac{2}{\delta}}{2mn} \right).
\]

### A.2 Proof of Theorem 1

Proof. The STL model aims to solve the following optimization problem as

\[
\min_{h_t^{\text{STL}}, f_t^{\text{STL}} \in \mathcal{F}} \frac{1}{mn} \sum_{t=1}^{m} \sum_{i=1}^{n} \mathcal{L}_t(y_i^t, h_t^{\text{STL}}(f_t^{\text{STL}}(x_i)))
\]

where its solution is denoted by \( \{h_t^{\text{STL}}\} \) and \( \{f_t^{\text{STL}}\} \). Therefore, the minimal empirical loss of the STL model on task \( t \) is computed as

\[
\hat{L}_t^{\text{STL}} = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_t(y_i^t, h_t^{\text{STL}}(f_t^{\text{STL}}(x_i))).
\]

For the DSMTL-IL model, we set \( \alpha_t = 0 \) for all tasks in the first training stage, thus the corresponding objective function is the same as that of the STL model. Therefore, the solution of the first stage in the DSMTL-IL model, i.e., \( \{h_t\} \) and \( \{f_t\} \), satisfies \( h_t = h_t^{\text{STL}} \) and \( f_t = f_t^{\text{STL}} \). In the second training stage of the DSMTL-IL model, for any public encoder \( f_s \) in task \( t \), we have

\[
\min_{\alpha_t \in \mathbb{M}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_t(y_i^t, h_t(\alpha_t f_s(x_i)) + (1 - \alpha_t) f_t(x_i))) \\
\leq \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_t(y_i^t, h_t^{\text{STL}}(f_t^{\text{STL}}(x_i))) = \hat{L}_t^{\text{STL}}.
\]

Therefore, \( L_t^* \leq \hat{L}_t^{\text{STL}} \leq L_t^{\text{STL}} \) holds for all \( 1 \leq t \leq m \). This finishes the proof.

### A.3 Proof of Theorem 2

Proof. According to Theorem 2 with \( m \) tasks and data \( X \), we have following inequality

\[
\frac{1}{m} \sum_{t=1}^{m} \hat{L}_t - \frac{1}{mn} \sum_{t,i} \mathcal{L}_t(y_i^t, h_t(g_t(f_s(x_i), f_t(x_i)))) \\
\leq \frac{c_1 G(\mathcal{F}(X))}{mn} + \frac{c_2 Q}{\sqrt{n}} + \left( \frac{9 \ln \frac{2}{\delta}}{2mn} \right).
\]

By substituting the solution of problem (3) into inequality (13), we have following inequality as

\[
\hat{L}_t - L_t^* \leq \frac{c_1 G(\mathcal{F}(X))}{n} + \frac{c_2 Q}{\sqrt{n}} + \left( \frac{9 \ln \frac{2}{\delta}}{2n} \right).
\]

Moreover, since the STL model can be considered as a special case of the DSMTL-IL model where \( g_t \) adopts \( g_t^0 \) defined in Eq. (5), substituting it into the inequality (13) gives

\[
L_t^{\text{STL}} - \mathcal{E}^{\text{STL}} \leq \frac{c_1 G(\mathcal{F}(X))}{mn} + \frac{c_2 Q}{\sqrt{n}} + \left( \frac{9 \ln \frac{2}{\delta}}{2mn} \right).
\]

Therefore, for the STL model in task \( t \), setting \( m \) to 1 in the above inequality gives

\[
L_t^{\text{STL}} - \mathcal{E}^{\text{STL}} \leq \frac{c_1 G(\mathcal{F}(X))}{n} + \frac{c_2 Q}{\sqrt{n}} + \left( \frac{18 \ln \frac{2}{\delta}}{n} \right).
\]

Add the inequalities (16) and (18) we get

\[
\hat{L}_t + \varepsilon_t \leq \mathcal{E}^{\text{STL}} + \frac{C_1 G(\mathcal{F}(X))}{n} + \frac{C_2 Q}{\sqrt{n}} + \left( \frac{18 \ln \frac{2}{\delta}}{n} \right),
\]

which completes the proof.

### A.4 Proof of Theorem 3

Proof. It is easy to see that the STL model can be considered as a special case of the DSMTL-JL model when \( \alpha_t = 0 \) holds for \( 1 \leq t \leq m \) and the HPS model is also a special case of the DSMTL-JL model when \( \alpha_t = 1 \) holds for \( 1 \leq t \leq m \).

The empirical loss of the DSMTL model is formulated as

\[
L = \frac{1}{mn} \sum_{t=1}^{m} \sum_{i=1}^{n} \mathcal{L}_t(y_i^t, h_t(g_t(f_s(x_i), f_t(x_i)))).
\]

We have \( L^{\text{STL}} = L(\Theta) \mid_{\alpha_t=0} \) and \( L^{\text{HPS}} = L(\Theta) \mid_{\alpha_t=1} \). Since \( L^* \leq L \), we can get \( L^* \leq \min\{L^{\text{STL}}, L^{\text{HPS}}\} \) and hence we reach the conclusion.
A.5 Proof of Theorem 4

Proof. Let $L^*$ be the optimal value of problem (4). Substituting the solution of problem (4) into inequality (11) gives

$$
\hat{\mathcal{E}} - L^* \leq \frac{C_1 G(F(\bar{X}))}{mn} + \frac{C_2 Q}{\sqrt{n}} + \sqrt{\frac{9 \ln \frac{2}{\delta}}{2mn}}.
$$

(19)

Based on inequalities (19) and (17), with probability $1 - \delta$, we have

$$
\hat{\mathcal{E}} - L^* + L_{\text{STL}} - \mathcal{E}_{\text{STL}} \leq \frac{C_1 G(F(\bar{X}))}{mn} + \frac{C_2 Q}{\sqrt{n}} + \sqrt{\frac{18 \ln \frac{2}{\delta}}{mn}},
$$

where $C_1$ and $C_2$ are two constants. According to Theorem 3, there exists a constant $\varepsilon \geq 0$ such that $L^* + \varepsilon = L_{\text{STL}}$. Therefore, we have

$$
\hat{\mathcal{E}} + \varepsilon \leq \mathcal{E}_{\text{STL}} + \frac{C_1 G(F(\bar{X}))}{mn} + \frac{C_2 Q}{\sqrt{n}} + \sqrt{\frac{18 \ln \frac{2}{\delta}}{mn}},
$$

where we reach the conclusion.

---

A.6 Proof of Theorem 5

Proof. Let $\{f_t\}^*, f_S^*, \{h_t\}^*, \{g_t\}^*$ be the minimizer in $\mathcal{E}^*$. We can decompose $\hat{\mathcal{E}} - \mathcal{E}^*$ as

$$
\hat{\mathcal{E}} - \mathcal{E}^* = \left(\frac{1}{mn} \sum_{t,i} \mathcal{L}_t(y^t_i, h^*_t(g^*_t(x^*_t), f^*_t(x^*_t))) \right)
+ \left(\frac{1}{mn} \sum_{t,i} \mathcal{L}_t(y^t_i, h_t(g_t(x^*_t), f^*_t(x^*_t))) \right)
- \left(\frac{1}{mn} \sum_{t,i} \mathcal{L}_t(y^t_i, h^*_t(g_t(x^*_t), f^*_t(x^*_t)) \right)
+ \left(\frac{1}{mn} \sum_{t,i} \mathcal{L}_t(y^t_i, h^*_t(g^*_t(x^*_t), f^*_t(x^*_t)) \right) - \mathcal{E}^*),
$$

where the first term can be bounded by substituting inequality (11) and the last term can be regarded as $mn$ random variables $\mathcal{L}_t(y^t_i, h^*_t(g^*_t(x^*_t), f^*_t(x^*_t))$ with values in $[0, 1]$. By using Hoeffding’s inequality, with probability at least $1 - \delta$, we have

$$
\frac{1}{mn} \sum_{t,i} \mathcal{L}_t(y^t_i, h^*_t(\omega_i^T \varphi^*_t(x_i^*_t))) - \mathcal{E}^* \leq \sqrt{\frac{\ln \frac{1}{\delta}}{2mn}}.
$$

The second term is non-positive due to the definition of minimizers. Therefore, we have

$$
\hat{\mathcal{E}} - \mathcal{E}^* \leq \frac{C_1 G(F(\bar{X}))}{mn} + \frac{C_2 Q}{\sqrt{n}} + \sqrt{\frac{8 \ln \frac{4}{\delta}}{mn}},
$$

where we reach the conclusion.