# MetaWeighting: Learning to Weight Tasks in Multi-Task Learning

**Yuren Mao**[1], **Zekai Wang**[2], **Weiwei Liu**[2*], **Xuemin Lin**[1], **Pengtao Xie**[3]
[1]School of Computer Science and Engineering, University of New South Wales
[2]School of Computer Science, Wuhan University
[3]Department of Electrical and Computer Engineering, University of California San Diego
yuren.mao@unsw.edu.au, {wzekai99,liuweiwei863}@gmail.com
lxue@cse.unsw.edu.au, pengtaoxie2008@gmail.com

## Abstract

Task weighting, which assigns weights on the including tasks during training, significantly matters the performance of Multi-task Learning (MTL); thus, recently, there has been an explosive interest in it. However, existing task weighting methods assign weights only based on the training loss, while ignoring the gap between the training loss and generalization loss. It degenerates MTL's performance. To address this issue, the present paper proposes a novel task weighting algorithm, which automatically weights the tasks via a learning-to-learn paradigm, referred to as MetaWeighting. Extensive experiments are conducted to validate the superiority of our proposed method in multi-task text classification.

## 1 Introduction

Multi-task Learning (MTL) simultaneously learns multiple related tasks and aims to achieve better performance than learning each task independently (Caruana, 1993; Baxter, 2000). It has achieved great success in various applications; especially, in the text classification context, MTL can significantly outperform single task learning (Liu et al., 2017; Mao et al., 2021).

In MTL, it is common for the including tasks to be competing. If we cannot properly balance these tasks, some tasks might dominate the training process and hurt the performance of other tasks, a phenomenon known as task imbalance. To address the task imbalance, the most widely used method is task weighting, which adaptively assigns weights on the tasks during training to balance their impacts. Various task weighting methods have been proposed and can be used in multi-task text classification, such as (Kendall et al., 2018; Sener and Koltun, 2018; Chen et al., 2018).

However, existing task weighting methods compute the task weights only based on training losses or corresponding gradients. They ignore the gap between the training loss and generalization loss. To illustrate this gap, we report observations of our four-task topic classification experiment in Figure 1. The detailed experimental settings are introduced in the experiment section. Figure 1 demonstrates that the training losses and generalization losses (estimated by the test losses) have different magnitudes; moreover, they have different patterns, such as a task might have the largest training loss but the lowest generalization loss among the tasks.

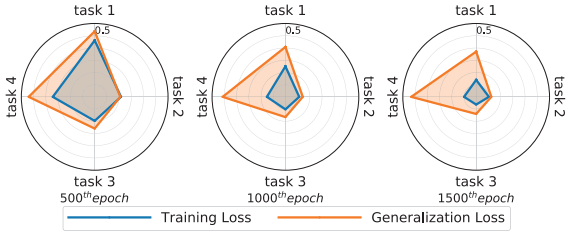


Figure 1: Illustration of the gap between training loss and generalization loss in the training process of a four-task topic classification experiment ($500^{th}$ , $1000^{th}$, $1500^{th}$ epochs respectively).

This gap causes a mismatch between the task weights and tasks' generalization performance, which reduces effectiveness of the task weighting. To tackle this issue, this paper proposes a novel task weighting method based on a bi-level optimization problem, which aims to find task weights that explicitly optimize the generalization performance. Our proposed method computes task weights by solving this bi-level optimization problem and performs in a learning-to-learn manner; thus, dubbed MetaWeighting. MetaWeighting can improve the performance of multi-task text classification.

To verify our theoretical analysis and validate the superiority of MetaWeighting, we conduct experiments on two classical text classification problems: sentiment analysis (on reviews) and topic classification (on news). The results demonstrate that MetaWeighting outperforms several state-of-the-art multi-task text classification methods.



*Findings of the Association for Computational Linguistics: ACL 2022*, pages 3436 - 3448

## 2 Related Works

Existing task weighting strategies can be divided into two categories: *weight adaptation methods* and *Pareto Optimization (PO)-based methods*. The *weight adaptation methods* adaptively adjust the tasks' weights during training based on pre-defined heuristic, such as uncertainty (Kendall et al., 2018), task difficulty prioritization (Guo et al., 2018), gradient normalization (Chen et al., 2018), weight average (Liu et al., 2019) and task variance regularization (Mao et al., 2021). These methods only use training losses or their gradients to compute task weights while ignores the gap between the training loss and generalization loss.

Besides, the *PO-based methods* formulate MTL as a multi-objective optimization problem and aim to find an arbitrary Pareto stationary solution (Sener and Koltun, 2018; Lin et al., 2019; Mahapatra and Rajan, 2020; Lin et al., 2020; Ma et al., 2020; Mao et al., 2020). However, in these methods, the learning objectives only involve training losses; thus, they can only achieve Pareto stationary points w.r.t training losses. They also ignore the gap between the training loss and generalization loss. Moreover, (Lin et al., 2019) proposes that the *PO-based methods* can be also regarded as weight adaptation methods for they optimize the weighted sum of training losses as well.

Overlooking the gap between the training loss and generalization loss would degenerate the performance of MTL. This paper proposes a novel meta learning-based task weighting method to solve this issue. There are some works adopt meta learning-based weighting methods in multilingual learning, e.g., (Wang et al., 2020) and (Tarunesh et al., 2021). However, these works cannot solve multi-objective optimization problems. By contrast, this paper proposes a novel method which can solve multi-objective optimization problems.

## 3 Preliminaries

Consider a multi-task learning problem with $T$ tasks over an input space $\mathcal{X}$ and a collection of task spaces $\{\mathcal{Y}_t\}_{t=1}^T$. For each task, we have a set of i.i.d. training samples $D_t = \{x_t^i, y_t^i\}_{i=1}^n$. The training samples are sampled from an identical distribution $\mathcal{P}_t$. Based on the training sets $\{D_t\}_{t=1}^T$, we learn an MTL model from a parameterized hypothesis class $\mathcal{H}$, which shares some parameters across tasks. Let $\theta_s$ represent the parameters shared between tasks (task-sharing parameters), while $\theta_t$ represent the task-specific parameters. $h(\cdot, \theta_s, \theta_1, ..., \theta_T) : \mathcal{X} \to \{\mathcal{Y}_t\}_{t=1}^T \in \mathcal{H}$ denotes an MTL model that learns from $\mathcal{H}$, while $h(\cdot, \theta_s, \theta_t) : \mathcal{X} \to \mathcal{Y}_t$ denotes the task-specific module in the MTL model.

The loss function is represented by $l(\cdot, \cdot) : \mathcal{Y}^t \times \mathcal{Y}^t \to [0, 1]^T$. For each task, the generalization loss is $\mathcal{L}_t(\theta) = \mathbb{E}_{(x_t,y_t)\sim\mathcal{P}_t} l(h(x_t, \theta_s, \theta_t), y_t)$, and the training loss is defined as $\mathcal{L}_t^{tr}(\theta, D_t) = \frac{1}{|D_t|}\sum_{(x_t,y_t)\in D_t} l(h(x_t, \theta_s, \theta_t), y_t)$. In this paper, each training set $D_t$ is randomly divided into two subsets: support set $D_t^s$ and query set $D_t^q$. Correspondingly; moreover, the support loss is defined as $\mathcal{L}_t^s(\theta, D_t^s) = \frac{1}{|D_t^s|}\sum_{(x_t,y_t)\in D_t^s} l(h(x_t, \theta_s, \theta_t), y_t)$, and the query loss is defined as $\mathcal{L}_t^q(\theta, D_t^q) = \frac{1}{|D_t^q|}\sum_{(x_t,y_t)\in D_t^q} l(h(x_t, \theta_s, \theta_t), y_t)$.

### 3.1 Hypergradient Descent

Hypergradient Descent (HD) (Almeida et al., 1998; Baydin et al., 2018) provides an efficient way to apply gradient descent on hyper-parameters. Here, we take learning rate's HD as an example to introduce the basic form of HD. Given an objective function $f(\theta)$ and previous parameters $\theta^{k-1}$, gradient descent-based learning typically evaluates the gradient $\nabla f(\theta^{k-1})$ and moves against it to arrive at updated parameters

$$\theta^k = \theta^{k-1} - \eta \nabla f(\theta^{k-1}), \tag{1}$$

where $\eta$ is the learning rate. HD derives an update rule for the learning rate $\eta$ itself. Based on Eq. (1) and the chain rule, we have

$$\begin{aligned} \frac{\partial f(\theta^k)}{\partial \eta} &= \nabla f(\theta^k) \cdot \frac{\partial(\theta^{k-1} - \eta \nabla f(\theta^{k-1}))}{\partial \eta} \\ &= \nabla f(\theta^k) \cdot (-\nabla f(\theta^{k-1})), \end{aligned} \tag{2}$$

with which we construct a update rule for $\eta$:

$$\eta^{k+1} = \eta^k + \beta \nabla f(\theta^k) \cdot \nabla f(\theta^{k-1}), \tag{3}$$

introducing $\beta$ as the hypergradient step size. In this paper, we extend HD to a bi-level multi-objective optimization problem.

### 3.2 Common Descent Direction for Multiple Objectives

When using gradient descent to jointly optimize multiple optimization objectives, we need to find a descent direction common to all the objectives. Based on the descent direction for each objective, (Désidéri, 2012) proposes a way to obtain the common descent direction, as in Theorem 1. This paper

proposes a method to simultaneously optimize the tasks' generalization loss based on Theorem 1.

**Theorem 1** ((Désidéri, 2012)). *Let $\mathcal{A}$ be a Hilbert space of finite or infinite dimension $N$. Let $f_i(z)$ $(1 \leq i \leq n \leq N)$ be $n$ smooth functions of the vector $z \in \mathcal{A}$. and $z^0$ a particular admissible design-point, at which the gradient-vectors are denoted $g_i = \nabla f_i(z^0)$, and*

$$\mathcal{U} = \{a \in \mathcal{A} | a = \sum_{i=1}^{n} \lambda_i g_i; \lambda_i > 0 (\forall i); \sum_{i=1}^{n} \lambda_i = 1\}. \quad (4)$$

*Let $a^* = \arg\min_{a \in \bar{\mathcal{U}}} \parallel a \parallel$, where $\mathcal{U}$ consists of the convex hull and closure of $\mathcal{U}$. Then, if $a^* \neq 0$, $a^*$ is a descent direction common to all the objectives.*

## 4 MetaWeighting for MTL

In this section, we demonstrate the gap between existing task weighting strategies and the generalization performance of MTL in Section 4.1. This gap motivates us to proposed a MetaWeighting problem, which aims to automatically learn a task weighting strategy that can narrow this gap, in Section 4.2. Moreover, we propose an algorithm to solve the MetaWeighting problem in Section 4.3.

### 4.1 Gap Between Task Weighting and Generalization Performance

MTL aims to improve the generalization performance of all the including tasks, which can be formulated via the following optimization problem.

$$\min_{\theta} \mathbf{L}(\theta) = (\mathcal{L}_1(\theta), ..., \mathcal{L}_T(\theta))^\top. \quad (5)$$

By contrast, existing task weighting strategies train an MTL model via the following objective.

$$\min_{\theta} \frac{1}{T} w_t \mathcal{L}_t^{tr}(\theta, D_t), \quad (6)$$

where the $w_t$ is adaptive during training and only depends on the training losses or their gradients. As the neural networks are usually heavily over-parameterized (Allen-Zhu et al., 2019), the training losses cannot properly estimate the generalization losses. Thus, existing task weighting strategies, which tunes weights only based on the training losses, overlook the generalization losses. Obviously, there is a gap between these task weighting strategies and the generalization performance of MTL.

### 4.2 MetaWeighting Problem

To narrow the gap between task weighting strategies and generalization performance, we propose to automatically learn task weights that can reduce the generalization losses, namely *learning to weight*. This *learning to weight* problem is formlated via the following bi-level optimization problem, dubbed MetaWeighting.

**Problem 1.**

$$\begin{aligned} &\min_{\boldsymbol{w}} \ (\mathcal{L}_1(\theta^*(\boldsymbol{w})), ..., \mathcal{L}_T(\theta^*(\boldsymbol{w})))^\top \\ &s.t. \quad \theta^*(\boldsymbol{w}) = \arg\min_{\theta} \frac{1}{T} \sum_{t=1}^{T} w_t \mathcal{L}_t^{tr}(\theta, D_t) \end{aligned} \quad (7)$$

where $\mathbf{w} = (w_1, w_2, ..., w_T)$. This bi-level optimization problem combines (5) and (6) together, by solving which we can obtain task weights that benefit the generalization performance of MTL.

However, the generalization loss is agnostic. To properly estimate the generalization loss, we randomly divide the training set $D_t$ into two subsets: support set $D_t^s$ and query set $D_t^q$, where $D_t^s$ is used to train an MTL model, and $D_t^q$ is used to estimate generalization loss of the MTL model. In Section 5, we theoretically demonstrate that query loss is a good estimator for the generalization loss; besides, in Section 6.7, experimental analysis also supports that query loss is a good estimator.

Based on the support-query split, the MetaWeighting problem is transformed into the following form.

**Problem 2.**

$$\begin{aligned} &\min_{\boldsymbol{w}} \ (\mathcal{L}_1^q(\theta^*(\boldsymbol{w}), D_1^q), ..., \mathcal{L}_T^q(\theta^*(\boldsymbol{w}), D_T^q))^\top \\ &s.t. \quad \theta^*(\boldsymbol{w}) = \arg\min_{\theta} \frac{1}{T} \sum_{t=1}^{T} w_t \mathcal{L}_t^{s}(\theta, D_t^s) \end{aligned} \quad (8)$$

### 4.3 MetaWeighting Algorithm

In the MetaWeighting problem, the inner optimization objective is embedded within the outer optimization objective. In MTL, the inner optimization objective is to minimize the weighted sum of task-specific training losses, which is typically optimized by means of iterative gradient descent; thus, Problem 2 can be formulated by the following problem in the $k^{th}$ learning iteration.

**Problem 3.**

$$\min_{\mathbf{w}} (\mathcal{L}_1^q(\theta^k, D_1^q), ..., \mathcal{L}_T^q(\theta^k, D_T^q))^\top$$
$$s.t. \quad \theta^k = \theta^{k-1} - \frac{\eta}{T}\sum_{t=1}^{T} w_t \nabla_\theta \mathcal{L}_t^s(\theta^{k-1}, D_t^s) \tag{9}$$

To solve Problem 3, we adopt the Hypergradient Descent (HD) method. However, the original HD method (Almeida et al., 1998; Baydin et al., 2018) is proposed for single objective optimization, which can not used in our problem where a multi-objective optimization problem involves. In this section, this paper proposes a novel HD method for the multi-objective optimization setting, as in the following sections.

#### 4.3.1 Task-Specific Descent Direction

The learning objective of Problem 3 involves $T$ objectives. We aim to find a gradient direction, moving against which all the objective can be optimized. To find this gradient direction, we first find the hypergradient direction w.r.t $\mathbf{w}$ (denoted as $d_t$) for each task. $d_t$ is computed by the following equation.

$$\begin{aligned} d_t &= \frac{\partial \mathcal{L}_t^q(\theta^k, D_t^q)}{\partial \mathbf{w}} = \nabla_\theta \mathcal{L}_t^q(\theta^k, D_t^q) \cdot \frac{\partial \theta^k}{\partial \mathbf{w}} \\ &= -\frac{\eta}{T} \nabla_\theta \mathcal{L}_t^q(\theta^k, D_t^q) \nabla_\theta \mathbf{L}^s(\theta^{k-1}, D^s). \end{aligned} \tag{10}$$

where $\nabla_\theta \mathbf{L}^s(\theta^{k-1}, D^s) = (\nabla_\theta \mathcal{L}_1^s(\theta^{k-1}, D_1^s)^\top, ..., \nabla_\theta \mathcal{L}_T^s(\theta^{k-1}, D_T^s)^\top)$. Moving against $d_t$, the generalization loss of task $t$ can be optimized.

#### 4.3.2 Common Descent Direction

Base on $d_t$, in this section, we find a common gradient direction, moving against which all the objective can be optimized. Let $\mathbf{d} = (d_1^\top, d_2^\top, ..., d_T^\top)$ and $d_c$ be the common gradient direction. Theorem 1 presents that the following Eq. (11) is a common descent direction.

$$d_c = \lambda^* \mathbf{d}^\top \tag{11}$$

where

$$\lambda^* = \arg\min_{\lambda} \{\| \lambda \mathbf{d}^\top \|_2^2 \, | \lambda \mathbf{1}^\top = 1, \lambda \succeq \mathbf{0}\}, \tag{12}$$

where $\mathbf{1} = (1, 1, ..., 1)$. Eq. (12) is a typical minimum Euclidean-norm point problem. We here adopt the widely used Frank-Wolfe optimization algorithm (Jaggi, 2013), a minimum-norm-point algorithm, to solve it. The Frank-Wolfe optimization algorithm is presented in Algorithm 2.

**Algorithm 1:** MetaWeighting Algorithm

**Input:** data $\{D_t^s\}_{t=1}^T$ and $\{D_t^q\}_{t=1}^T$, Number of learning iterations $K$, step size $\alpha$ for updating $\mathbf{w}$.
**Initialize:** $w^0 = (1, 1, ..., 1)$, $\theta^0$, $\eta$.
**for** $k = 1$ **to** $K$ **do**
  $\theta^k = \theta^{k-1} - \frac{\eta}{T}\sum_{t=1}^{T} w_t \nabla_\theta \mathcal{L}_t^s(\theta^{k-1}, D_t^s)$.
  **for** $t = 1$ **to** $T$ **do**
    $d_t = -\frac{\eta}{T} \nabla_\theta \mathcal{L}_t^q(\theta^k, D_t^q) \nabla_\theta \mathbf{L}^s(\theta^{k-1}, D^s)$.
  **end for**
  $\mathbf{d} = ({d_1}^\top, {d_2}^\top, ..., {d_T}^\top)$
  $\lambda^* = \arg\min_\lambda \{\| \lambda \mathbf{d}^\top \|_2^2 \, | \lambda \mathbf{1}^\top = 1, \lambda \succeq \mathbf{0}\}$ (calls Algorithm 2).
  $d_c = \lambda^* \mathbf{d}^\top$.
  $\mathbf{w}^{k+1} = \mathbf{w}^k - \alpha d_c$.
**end for**

**Algorithm 2:** Frank-Wolfe Algorithm

**Input:** Number of Iterations $N$.
**Initialize:** $\lambda_0 = [\frac{1}{T}, ..., \frac{1}{T}]$.
$B = \mathbf{d}^\top \mathbf{d}$.
**for** $i = 0$ **to** $N$ **do**
  $v = \arg\min_{v \in \{v^\top \mathbf{1} = 1, v \succeq \mathbf{0}\}} v^\top B \lambda$.
  $\gamma = \arg\min_{\gamma \in [0,1]} (\lambda_i + \gamma(v - \lambda_i))^\top B(\lambda_i + \gamma(v - \lambda_i))$.
  $\lambda_{i+1} = (1 - \gamma)\lambda_i + \gamma v$.
**end for**
**return:** $\lambda_N$

#### 4.3.3 MetaWeighting

Moving against $d_c$, all the objective can be optimized; thus, the update rule of $\mathbf{w}$ is

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \alpha d_c, \tag{13}$$

where $\alpha$ is the step size. Based on this update rule, the task weights are automatically learnt oriented by optimizing the generalization losses.

Overall, we propose the MetaWeighting algorithm, which is presented in algorithmic form in Algorithm 1. Our proposed method bridges the gap between task weighting and generalization performance of MTL.

## 5 Theoretical Analysis

In this section, we study the generalization error bound for MTL; furthermore, we compare the bound w.r.t training loss and the bound w.r.t the

query loss. The comparison presents that the query loss is a more accurate estimation of the generalization loss than the training loss.

Firstly, we derive the generalization error bound w.r.t training loss for MTL.

**Theorem 2.** *Assume we have $n$ training samples for each task. Let $\sigma = \{\{\sigma_i^t\}_{i=1}^n\}_{t=1}^T$ be a sequence of binary random variables such that each $\sigma_i^t = \pm 1$ is independent with probability $1/2$. Then, $\forall \delta \in [0,1]$, for all $h(\cdot, \theta^s, \theta^1, ..., \theta^T) \in \mathcal{H}$, with probability of at least $1-\delta$:*

$$\frac{1}{T}\sum_{t=1}^{T} (\mathcal{L}_t(\theta) - \mathcal{L}_t^{tr}(\theta, D_t)) \leq 2R(l \circ \mathcal{H} \circ D) + 4\sqrt{\frac{2\log(4/\delta)}{Tn}}. \tag{14}$$

*where*

$$R(l \circ \mathcal{H} \circ D) = \mathbb{E}_\sigma \sup_\theta \left(\frac{1}{Tn}\sum_{t=1}^{T}\sum_{i=1}^{n} \sigma_i^t l(h(x_i^t, \theta), y_i^t)\right). \tag{15}$$

*is the Rademacher complexity for MTL.*

*Proof.* The proof is provided in Appendix A. □

Next, we derive the generalization error bound w.r.t query loss for MTL.

**Theorem 3.** *Assume we have $m$ training samples for each task. $\forall \delta \in [0,1]$, with probability of at least $1-\delta$, for all $h(\cdot, \theta^s, \theta^1, ..., \theta^T) \in \mathcal{H}$, we have*

$$\frac{1}{T}\sum_{t=1}^{T}(\mathcal{L}_t(\theta) - \mathcal{L}_t^q(\theta, D_t^q)) \leq \sqrt{\frac{log(2/\delta)}{2m}}. \tag{16}$$

*Proof.* The proof is provided in Appendix A. □

Comparing the bound (14) and (16), we can find that the upper bound for the query loss is tighter than that for the training loss. Taking $m$ to be order of $n$, the query loss is a more accurate estimate of the generalization loss than the training loss by a factor that depends on the Rademacher complexity.

## 6 Experiments

In this section, we perform experimental studies on sentiment analysis to evaluate the performance of our proposed MetaWeighting and verify our theoretical analysis.

### 6.1 Datasets

**Sentiment Analysis** [1]. We evaluate our algorithm on product reviews from Amazon. The dataset (Blitzer et al., 2007) contains product reviews from 14 domains, including books, DVDs, electronics, kitchen appliances and so on. We consider each domain as a binary classification task. Reviews with rating $> 3$ were labeled positive, those with rating $< 3$ were labeled negative, reviews with rating $= 3$ are discarded as the sentiments were ambiguous and hard to predict.

**Topic Classification** [2]. We select 16 newsgroups from the 20 Newsgroup dataset, which is a collection of approximately 20,000 newsgroup documents that is partitioned (nearly) evenly across 20 different newsgroups, then formulate them into four 4-class classification tasks (as shown in Table 1) to evaluate the performance of our algorithm on topic classification.

Table 1: Data Allocation for Topic Classification Tasks.

| TASKS | NEWSGROUPS |
|---|---|
| COMP | OS.MS-WINDOWS.MISC, SYS.MAC.HARDWARE, GRAPHICS, WINDOWS.X |
| REC | SPORT.BASEBALL, SPORT.HOCKEY AUTOS, MOTORCYCLES |
| SCI | CRYPT, ELECTRONICS, MED, SPACE |
| TALK | POLITICS.MIDEAST, RELIGION.MISC, POLITICS.MISC, POLITICS.GUNS |

### 6.2 Baselines

We compare MetaWeighting with methods:

**Single-Task Learning (STL):** learning each task independently.

**Uniform:** learning tasks simultaneously using uniform task weights.

**Uncertainty:** using the uncertainty weighting method proposed by (Kendall et al., 2018).

**GradNorm:** using the gradient normalization method proposed by (Chen et al., 2018).

**MGDA:** using the MGDA-UB method proposed by (Sener and Koltun, 2018).

**AdvMTL:** using the adversarial Multi-task Learning method proposed by (Liu et al., 2017).

**TchebycheffAdv:** using the Adversarial Tchebycheff procedure proposed by (Mao et al., 2020).

**BanditMTL:** using the BanditMTL method proposed by (Mao et al., 2021).

[1] https://www.cs.jhu.edu/~mdredze/datasets/sentiment/

[2] http://qwone.com/~jason/20Newsgroups/

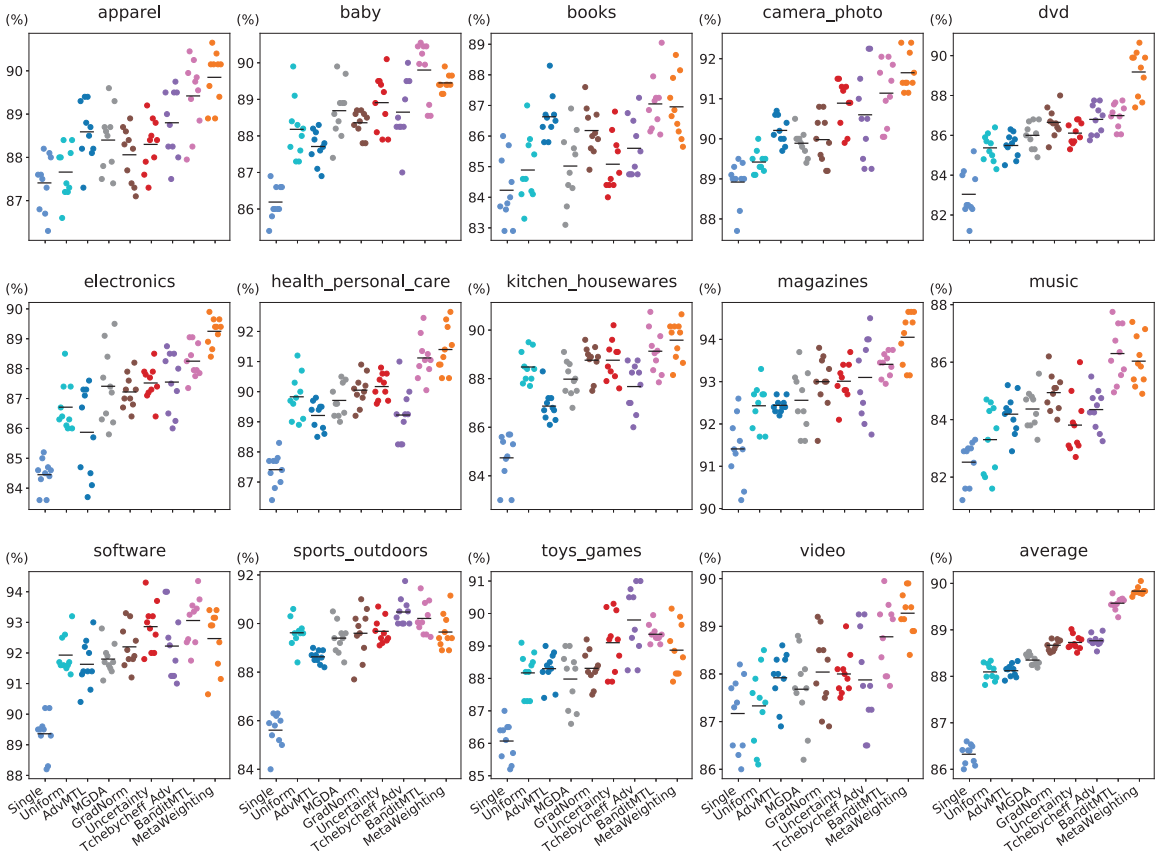


Figure 2: Classification accuracy of Single Task Learning, Uniform Scaling, AdvMTL, MGDA, GradNorm, Uncertainty, TchebycheffAdv, BanditMTL and MetaWeighting on TextCNN for the sentiment analysis dataset. Each colored cluster illustrates the classification accuracy performance of a method over 10 runs. Our proposed MetaWeighting outperforms all baselines on ten of the fourteen tasks; besides, its average performance is superior to that of all baselines.

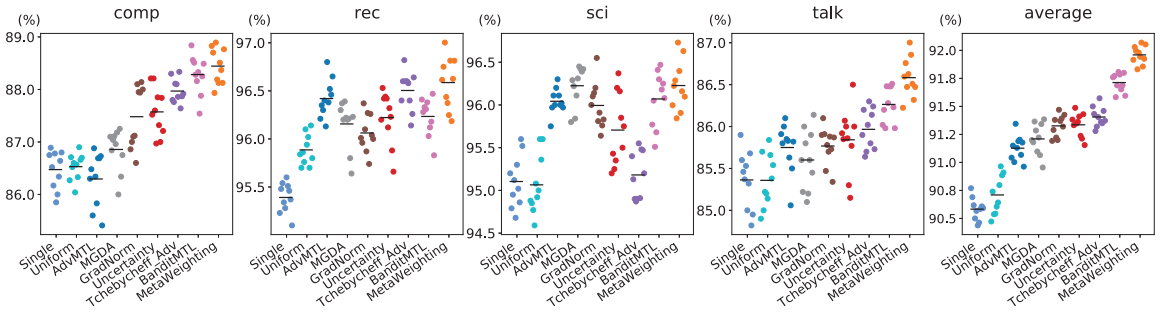


Figure 3: Classification accuracy of Single Task Learning, Uniform Scaling, AdvMTL, MGDA, GradNorm, Uncertainty, TchebycheffAdv, BanditMTL and MetaWeighting on TextCNN for the topic classification dataset. Each colored cluster illustrates the classification accuracy performance of a method over 10 runs. Our proposed MetaWeighting outperforms all baselines in all tasks.

### 6.3 Experimental Settings

We adopt the hard parameter-sharing MTL framework (Mao et al., 2021), where the shared representation extractor is built with TextCNN or BERT; besides, the task-specific module is formulated by means of one fully connected layer ending with a softmax function. The detailed experimental settings are introduced in the Appendix B.

### 6.4 Classification Performance

We compare the proposed MetaWeighting with the baselines and report the results over 10 runs by plotting the classification accuracy of each task for both sentiment analysis and topic classification. The results on TextCNN are shown in Fig. 2 and 3. Due to space limitations, we provide the results for BERT in the Appendix C. All experimental

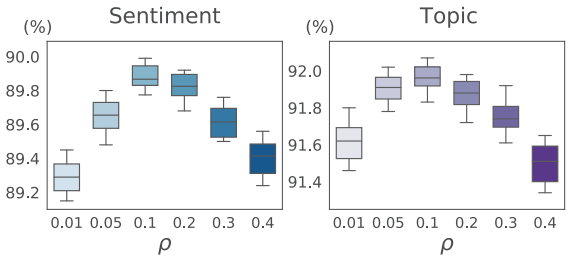


Figure 4: Task-average classification accuracy w.r.t different value of $\rho$ (query-split radio) for sentiment analysis and topic classification.

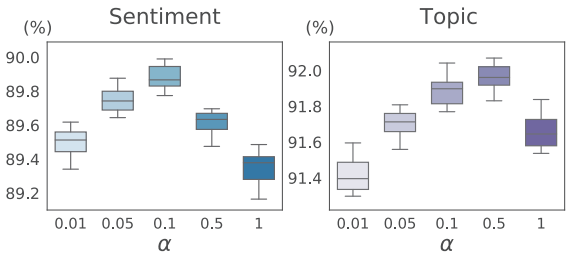


Figure 5: Task-average classification accuracy w.r.t different value of $\alpha$ (step size) for sentiment analysis and topic classification.

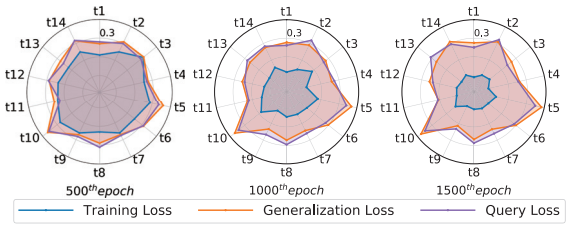


Figure 6: Illustration of the gap between training loss, query loss and generalization loss in the training process of sentiment analysis ($500^{th}$ , $1000^{th}$, $1500^{th}$ epochs respectively).

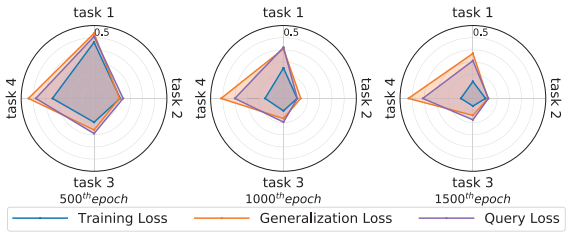


Figure 7: Illustration of the gap between training loss, query loss and generalization loss in the training process of topic classification ($500^{th}$ , $1000^{th}$, $1500^{th}$ epochs respectively).

results show that our proposed MetaWeighting outperforms all baselines and achieves state-of-the-art performance.

### 6.5 The Impact of Query-Split Radio

Let $n$ be the size of the entire training set and $m$ be the size of the query set. We define the query-split radio as $\rho = \frac{m}{n}$ to indicate the radio of query samples to the entire training samples. From the theoretical analysis of Section 5, we can see that the query loss can estimate generalization loss more accurately when $\rho$ increases, but increasing $\rho$ would hurt the training process for the size of support set decreases. Therefore, $\rho$ faces a trade-off between the performance estimation of generalization loss and training performance.

To investigate the impact of $\rho$, we record the changes in MetaWeighting's average classification accuracy w.r.t different values of $\rho$ in Fig. 4, where each boxplot visually illustrates the distribution of results over ten runs through displaying the data quartiles (first quartile and third quartile), minimum/maximum value and median. These experiments are conducted based on TextCNN. In this figure, as $\rho$ increases, the average accuracy of MetaWeighting first increases and then decreases. It verifies our theoretical analysis. For both sentiment analysis and topic classification, setting $\rho = 0.1$ provides satisfactory results.

### 6.6 Sensitive Study on $\alpha$

In MetaWeighting, the step size $\alpha$ is a hyper-parameter. To determine whether the performance of MetaWeighting is sensitive to $\alpha$, we conduct experiments on the classification accuracy performance of MetaWeighting w.r.t different values of $\alpha$ based on the TextCNN model. The results of these experiments are presented in Figure 5 (boxplots over ten runs). As the figure shows, the performance of our proposed method is not very sensitive to $\alpha$ when $\alpha$ is within the range of $0.05$ to $0.1$ for sentiment analysis and $0.1$ to $0.5$ for topic classification. The results demonstrate that MetaWeighting can work well in a wide range of $\alpha$ values.

### 6.7 The Gap between the Training Loss, Query Loss and Generalization Loss

To experimentally verify that the query loss is a good estimator for generalization loss, we record the generalization loss (estimated by test loss), query loss and training loss for each task during training and report the results in Fig. 6 and 7 for sentiment analysis and topic classification respectively. From these figures, we can see that there

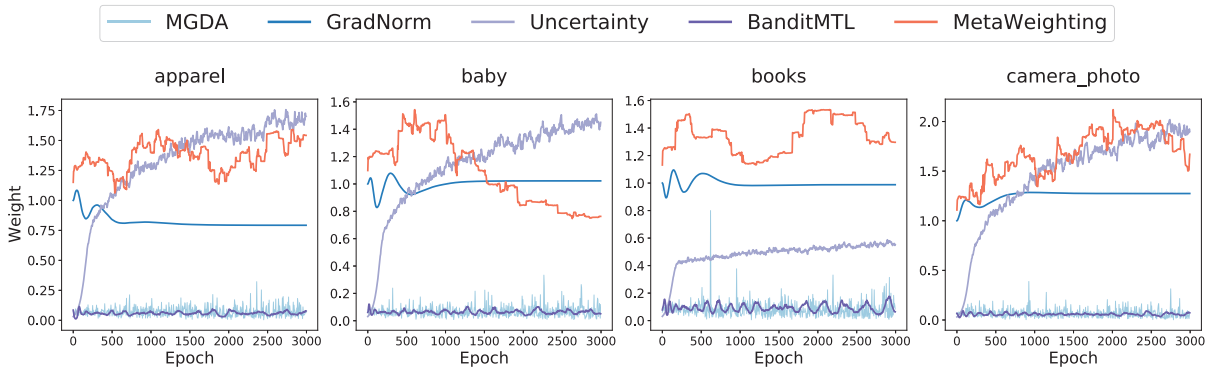


Figure 8: Comparison of task weight adaption processes between MetaWeighting, Uncertainty, Gradnorm, MGDA and BanditMTL for sentiment analysis.

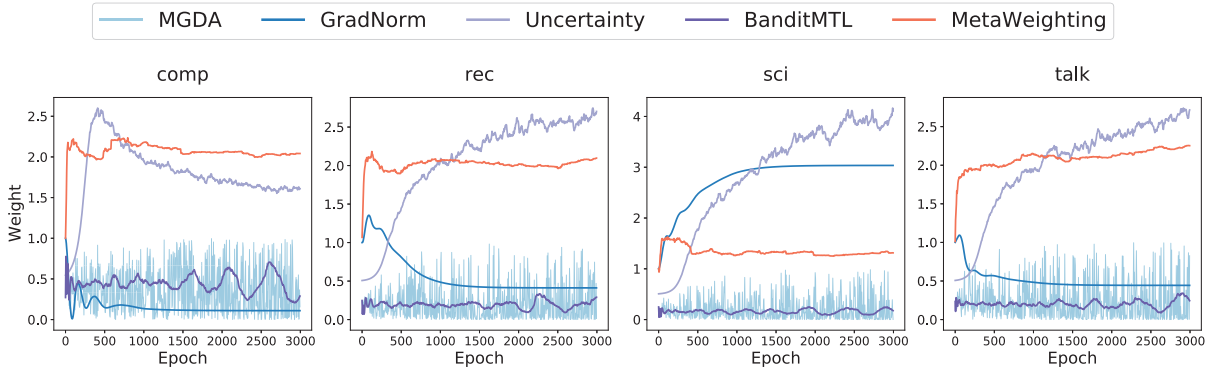


Figure 9: Comparison of task weight adaption processes between MetaWeighting, Uncertainty, Gradnorm, MGDA and BanditMTL for topic classification.

is a large gap between the training and generalization loss, while the gap between the query and generalization loss is smaller than that between the training and generalization loss. The results verify our theoretical analysis in Section 5; furthermore, they experimentally support our motivation for MetaWeighting.

In this section, TextCNN is used, and tasks have uniform weights during training. Fig. 1 is obtained under this setting as well.

### 6.8 The Evolution of Task Weights

In this section, we observe the changes in task weights in the training process of MetaWeighting and compare these changes with four baselines (Uncertainty, Gradnorm, MGDA and BanditMTL). The results for sentiment analysis and topic classification are reported in Fig. 8 and 9 respectively. Due to space limitations, for sentiment analysis, we only report the results of the first four tasks here, and the results of the other ten tasks are presented in the Appendix D.

From these figures, we can see that the weight adaption process of MetaWeighting is different with that of Uncertainty, Gradnorm, MGDA and BanditMTL. In MetaWeighting, the task weights are automatically learnt, and there is no pre-defined heuristic involved. It is verified by the evolution curves of task weights for MetaWeighting illustrated in Fig. 8 and 9, which fluctuate without any regular patterns.

## 7 Conclusion

This paper presents that the gap between the training loss and the generalization loss, which is overlooked by existing task weighting methods, is non-negligible; furthermore, to narrow this gap, a novel task weighting method (dubbed MetaWeighting) is proposed. In MetaWeighting, multi-task text classification is formulated as a multi-objective bi-level programming problem, and then solved in a learning-to-learn manner. MetaWeighting automatically learns the task weights without any pre-defined heuristic and achieves state-of-the-art performance. It has the potential to forge new trends in task weighting research.

## References


Jon Wellner Aad van der Vaart. 1996. *Weak convergence and empirical processes*. Springer.

Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. 2019. Learning and generalization in overparameterized neural networks, going beyond two layers. In *NeurIPS*.

Luís B Almeida, Thibault Langlois, José D Amaral, and Alexander Plakhov. 1998. Parameter adaptation in stochastic optimization. In *On-Line Learning in Neural Networks*, pages 111–134. Cambridge University Press.

Jonathan Baxter. 2000. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198.

Atilim Gunes Baydin, Robert Cornish, David Martínez-Rubio, Mark Schmidt, and Frank Wood. 2018. Online learning rate adaptation with hypergradient descent. In *ICLR*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *ICML*.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*.

Jean-Antoine Désidéri. 2012. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318.

Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. 2018. Dynamic task prioritization for multitask learning. In *ECCV*.

Martin Jaggi. 2013. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML*.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Xi Lin, Zhiyuan Yang, Qingfu Zhang, and Sam Kwong. 2020. Controllable pareto multi-task learning. *CoRR*.

Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qingfu Zhang, and Sam Kwong. 2019. Pareto multi-task learning. In *NIPS*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *ACL*.

Shikun Liu, Edward Johns, and Andrew J. Davison. 2019. End-to-end multi-task learning with attention. In *CVPR*.

Pingchuan Ma, Tao Du, and Wojciech Matusik. 2020. Efficient continuous pareto exploration in multi-task learning. In *ICML*.

Debabrata Mahapatra and Vaibhav Rajan. 2020. Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto optimization. In *ICML*.

Yuren Mao, Zekai Wang, Weiwei Liu, Xuemin Lin, and Wenbin Hu. 2021. Banditmtl: Bandit-based multi-task learning for text classification. In *ACL*.

Yuren Mao, Shuang Yun, Weiwei Liu, and Bo Du. 2020. Tchebycheff procedure for multi-task text classification. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. In *NeurIPS*.

Ishan Tarunesh, Sushil Khyalia, Vishwajeet Kumar, Ganesh Ramakrishnan, and Preethi Jyothi. 2021. Meta-learning for effective multi-task and multilingual modelling. In *EACL*.

Xinyi Wang, Yulia Tsvetkov, and Graham Neubig. 2020. Balancing training for multilingual neural machine translation. In *ACL*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP*.

## A Proof of the Theorem 2 and Theorem 3

**Lemma 1** (McDiarmid's Inequality)**.** *Let $V$ be some set and let $f : V^n \to \mathbb{R}$ be a function of $n$ variables such that for some $c > 0$ , for all $i \in [n]$ and for all $z_1, ..., z_n, z_i' \in V$ we have*

$$|f(z_1, ..., z_n) - f(z_1, ..., z_{i-1}, z_i', z_{i+1}, ..., z_n)| \leq c. \tag{17}$$

*Let $Z_1, ..., Z_n$ be $n$ independent random variables taking values in $V$. Then, with probability of at least $1 - \delta$ we have*

$$|f(Z_1, ..., Z_n) - \mathbb{E}[f(Z_1, ..., Z_n)]| \leq c\sqrt{\frac{n\log(2/\delta)}{2}}. \tag{18}$$

**Lemma 2** (Hoeffding's Inequality)**.** *Let $z_1, ..., z_m$ be a a sequence of i.i.d. random variables and assume that for all $i$, $\mathbb{E}(z_i) = \mu$ and $P(a \leq z_i \leq b) = 1$. Then, for any $\epsilon > 0$*

$$P\left[\left|\frac{1}{m}\sum_{i=1}^{m} z_i - \mu\right| > \epsilon\right] \leq 2exp(\frac{-2m\epsilon^2}{(b-a)^2}). \tag{19}$$

**Lemma 3.** *Assume that $\forall (x_t^i, y_t^i), (x_t^j, y_t^j)$ : $|l(h(x_t^i, \theta^s, \theta^t), y_t^i) - l(h(x_t^j, \theta^s, \theta^t), y_t^j)| \leq c$. Let*

$$Rep(\mathcal{H}, D) = \sup_{h\in\mathcal{H}} \frac{1}{T}\sum_{t=1}^{T}(\mathcal{L}_t(\theta) - \mathcal{L}_t^{tr}(\theta, D_t)), \tag{20}$$

*then $\forall \delta \in [0, 1]$, with probability of at least $1 - \delta$:*

$$Rep(\mathcal{H}, D) \leq \mathbb{E}_D Rep(\mathcal{H}, D) + c\sqrt{\frac{2\log(2/\delta)}{Tn}}. \tag{21}$$

*Proof.* Let $s_t^i = (x_t^i, y_t^i)$. The training set for MTL is $D = \{\{(s_1^1, ..., s_1^n\}, ..., \{s_t^1, ..., s_t^n\}, ..., \{s_T^1, ..., s_T^n\}\}$. For $\forall t, i$, replace $s_t^i$ with $u_t^i = (x_t^*, y_t^*) \in D_t$ and create a new dataset $\overline{D} = \{\{(s_1^1, ..., s_1^n\}, ..., \{s_t^1, ..., u_t^i, ..., s_t^n\}, ..., \{s_T^1, ..., s_T^n\}\}$. Let $h_t(\cdot) = h(\cdot, \theta^s, \theta^t)$. As $\forall (x_t^i, y_t^i), (x_t^j, y_t^j)$ : $|l(h(x_t^i, \theta^s, \theta^t), y_t^i) - l(h(x_t^j, \theta^s, \theta^t), y_t^j)| \leq c$, we have

$$\begin{aligned} &Rep(\mathcal{H}, D) - Rep(\mathcal{H}, \overline{D}) \\ &\leq \sup_{h\in\mathcal{H}} \tfrac{1}{Tn}|l(h_t(x_t^n), y_t^n) - l(h_t(x_t^*), y_t^*))| \leq \tfrac{c}{Tn}. \end{aligned} \tag{22}$$

Using the McDiarmid's Inequality (Lemma 1), we have

$$\begin{aligned} Rep(\mathcal{H}, D) &\leq \mathbb{E}_D Rep(\mathcal{H}, D) + \tfrac{2c}{Tn}\sqrt{\tfrac{Tn\log(2/\delta)}{2}} \\ &= \mathbb{E}_D Rep(\mathcal{H}, D) + c\sqrt{\tfrac{2\log(2/\delta)}{Tn}}. \end{aligned} \tag{23}$$

We conclude our proof. □

**Proof of Theorem 2**.

*Proof.* Using the standard symmetrization argument (for example, see Lemma 2.3.1 of (Aad van der Vaart, 1996) ), we have

$$\mathbb{E}_D Rep(\mathcal{H}, D) \leq 2\mathbb{E}_D R(l \circ \mathcal{H} \circ D). \tag{24}$$

Combining Eq. (21) and Eq. (24), with probability $1 - \delta/2$:

$$\begin{aligned} &\sup_{h\in\mathcal{H}} \tfrac{1}{T}\textstyle\sum_{t=1}^{T}(\mathcal{L}_t(\theta) - \mathcal{L}_t^{tr}(\theta, D_t)) \\ &\leq 2\mathbb{E}_D R(l \circ \mathcal{H} \circ D) + c\sqrt{\tfrac{2\log(4/\delta)}{Tn}}. \end{aligned} \tag{25}$$

Obviously, with probability of at least $1 - \delta/2$, for all $h \in \mathcal{H}$, we have

$$\begin{aligned} &\tfrac{1}{T}\textstyle\sum_{t=1}^{T}(\mathcal{L}_t(\theta) - \mathcal{L}_t^{tr}(\theta, D_t)) \\ &\leq 2\mathbb{E}_D R(l \circ \mathcal{H} \circ D) + c\sqrt{\tfrac{2\log(4/\delta)}{Tn}}. \end{aligned} \tag{26}$$

Let $s_t^i = (x_t^i, y_t^i)$. The training set for MTL is $D = \{\{(s_1^1, ..., s_1^n\}, ..., \{s_t^1, ..., s_t^n\}, ..., \{s_T^1, ..., s_T^n\}\}$.

For $\forall t, i$, replace $s_t^i$ with $u_t^i = (x_t^*, y_t^*) \in D_t$ and create a new dataset $\overline{D} = \{\{(s_1^1, ..., s_1^n\}, ..., \{s_t^1, ..., u_t^i, ..., s_t^n\}, ..., \{s_T^1, ..., s_T^n\}\}$.

Let $h_t(\cdot) = h(\cdot, \theta^s, \theta^t)$. As $\forall (x_t^i, y_t^i), (x_t^j, y_t^j)$ : $|l(h(x_t^i, \theta^s, \theta^t), y_t^i) - l(h(x_t^j, \theta^s, \theta^t), y_t^j)| \leq c$, we have

$$\begin{aligned} &Rep(\mathcal{H}, D) - Rep(\mathcal{H}, \overline{D}) \leq \\ &\sup_{h\in\mathcal{H}} \tfrac{1}{Tn}|l(h_t(x_t^n), y_t^n) - l(h_t(x_t^*), y_t^*))| \leq \tfrac{c}{Tn} \end{aligned} \tag{27}$$

Using the McDiarmid's Inequality (Lemma 1), we have that: with probability of at least $1 - \delta/2$:

$$\mathbb{E}_D R(l\circ\mathcal{H}\circ D) \leq R(l\circ\mathcal{H}\circ D) + 2c\sqrt{\tfrac{2\log(4/\delta)}{Tn}}. \tag{28}$$

Based on Eq. (28) and the union bound, we have that - with probability of at least $1 - \delta$:

$$\begin{aligned} &\tfrac{1}{T}\textstyle\sum_{t=1}^{T}(\mathcal{L}_t(\theta) - \mathcal{L}_t^{tr}(\theta, D_t)) \\ &\leq 2R(l \circ \mathcal{H} \circ D) + 4c\sqrt{\tfrac{2\log(4/\delta)}{Tn}}. \end{aligned} \tag{29}$$

In our setting, $l(\cdot,\cdot) : \mathcal{Y}^t \times \mathcal{Y}^t \to [0,1]$, then $c = 1$. We have

$$\begin{aligned} &\frac{1}{T}\sum_{t=1}^{T}(\mathcal{L}_t(\theta) - \mathcal{L}_t^{tr}(\theta, D_t)) \\ &\leq 2R(l \circ \mathcal{H} \circ D) + 4\sqrt{\tfrac{2\log(4/\delta)}{Tn}}. \end{aligned} \tag{30}$$

We conclude our proof. □

Based on the Hoeffding's Inequality (Lemma 2), we have the following theorem.

**Proof of Theorem 3**.

*Proof.* Based on the Hoeffding's Inequality (Lemma 2) and $l(\cdot,\cdot) : \mathcal{Y}^t \times \mathcal{Y}^t \to [0,1]$, for each $h(\cdot, \theta^s, \theta^t) \in \mathcal{H}^t$, we have

$$P\left[|\mathcal{L}_t(\theta) - \mathcal{L}_t^q(\theta, D_t)| > \epsilon\right] \leq 2exp(-2m\epsilon^2). \tag{31}$$

Then, with probability of at least $1-2exp(-2m\epsilon^2)$, we have

$$|\mathcal{L}_t(\theta) - \mathcal{L}_t^q(\theta, D_t)| \leq \epsilon. \tag{32}$$

Let $\delta = 2exp(-2m\epsilon^2)$, we have that with probability of at least $1-\delta$,

$$|\mathcal{L}_t(\theta) - \mathcal{L}_t^q(\theta, D_t)| \leq \sqrt{\frac{log(2/\delta)}{2m}}. \tag{33}$$

Thus, for each task,

$$\mathcal{L}_t(\theta) - \mathcal{L}_t^q(\theta, D_t) \leq \sqrt{\frac{log(2/\delta)}{2m}}. \tag{34}$$

Since the bound for each task are independent, we have

$$\frac{1}{T}\sum_{t=1}^{T}(\mathcal{L}_t(\theta) - \mathcal{L}_t^q(\theta, D_t)) \leq \sqrt{\frac{log(2/\delta)}{2m}}. \tag{35}$$

We conclude our proof. □

## B Detailed Experimental Settings

We adopt the hard parameter-sharing MTL framework (Mao et al., 2021), where the shared representation extractor is built with TextCNN or BERT; besides, the task-specific module is formulated by means of one fully connected layer ending with a softmax function. The TextCNN module is structured with three parallel convolutional layers with kernels size of 3, 5, 7 respectively. For TextCNN, we adopt Pre-trained GloVe (Pennington et al., 2014) word embeddings. By contrast, the BERT module is formulated via a pre-trained BERT-base model provided by Hugging Face(Wolf et al., 2020), with a hidden size of 768, 12 Transformer blocks and 12 self-attention heads.

We train the deep MTL network model in line with Algorithm 1. We set $\alpha$ to be 0.1 and 0.5 for sentiment analysis and topic classification respectively, and the query-split radio (radio of query samples to entire training samples) to be 0.1 for both sentiment analysis and topic classification. We use the Adam optimizer (Kingma and Ba, 2015). We train over 3000 epochs for TextCNN and fine-tune over 50 epochs for BERT. For TextCNN, the learning rate is $1e-3$ and the batch size is 256. For BERT, the learning rate is $2e-5$ , the batch size is 32, and the max sequence length is 256. For the baselines, we search over the set $\{1e{-}5, 2e{-}5, 5e{-}5, 1e{-}4, 5e{-}4, 1e{-}3, 5e{-}3\}$ learning rates and choose the model with best performance.

## C Classification Performance on BERT

For the BERT-based MTL model, we compare the proposed MetaWeighting with the baselines and report the results over 10 runs by plotting the classification accuracy of each task for both sentiment analysis and topic classification in Fig. 10 and 11. AdvMTL and TchebycheffAdv are not available for BERT; thus, we do not compare with AdvMTL and compare with Tchebycheff which is TchebycheffAdv without aversarial module (Mao et al., 2021). From these figures, we can see that our proposed MetaWeighting outperforms all baselines and achieves state-of-the-art performance.

## D The Evolution of Task Weights for Sentiment Analysis

Fig. 12 illustrates the changes in task weights in the training process of MetaWeighting for all the tasks of sentiment analysis.

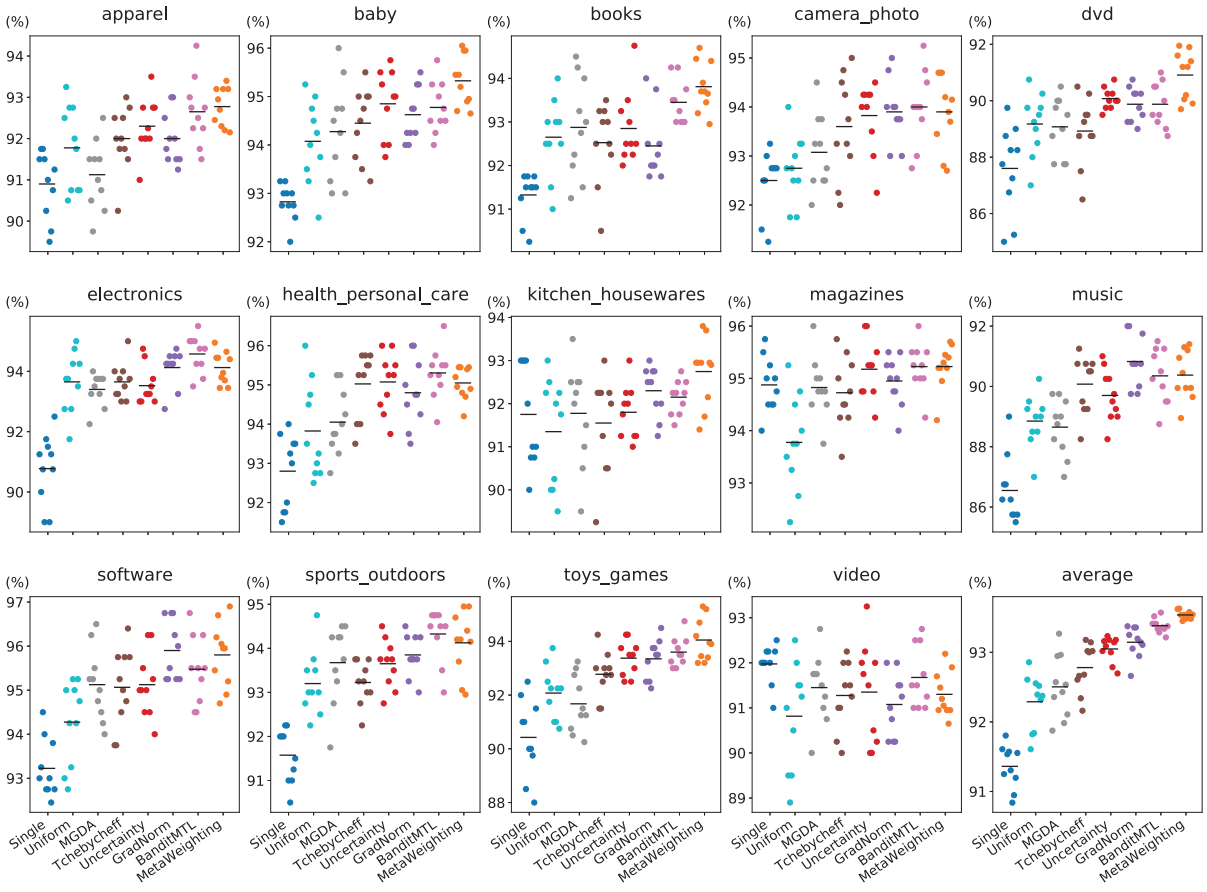


Figure 10: Classification accuracy of Single Task Learning, Uniform Scaling, MGDA, TchebycheffAdv, Uncertainty, GradNorm, BanditMTL and MetaWeighting on BERT for the sentiment analysis dataset. Each colored cluster illustrates the classification accuracy performance of a method over 10 runs. Our proposed MetaWeighting outperforms all baselines on eleven of the fourteen tasks; besides, its average performance is superior to that of all baselines.

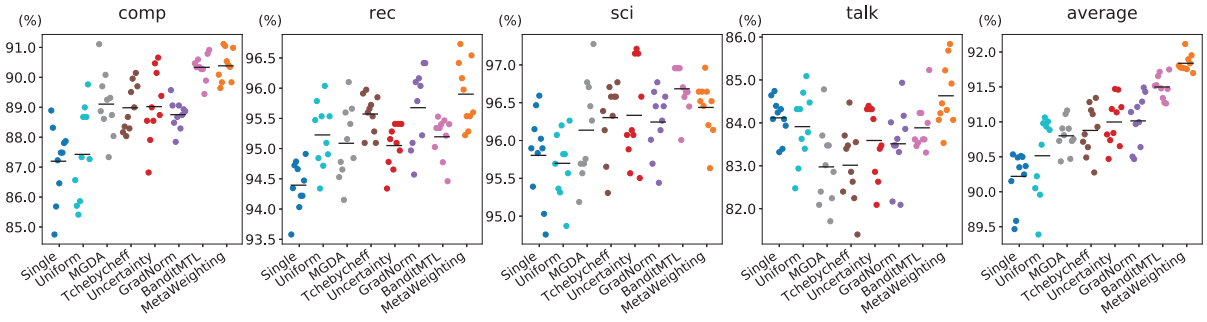


Figure 11: Classification accuracy of Single Task Learning, Uniform Scaling, MGDA, TchebycheffAdv, Uncertainty, GradNorm, BanditMTL and MetaWeighting on BERT for the topic classification dataset. Each colored cluster illustrates the classification accuracy performance of a method over 10 runs. Our proposed MetaWeighting outperforms all baselines on three of the four tasks; besides, its average performance is superior to that of all baselines.

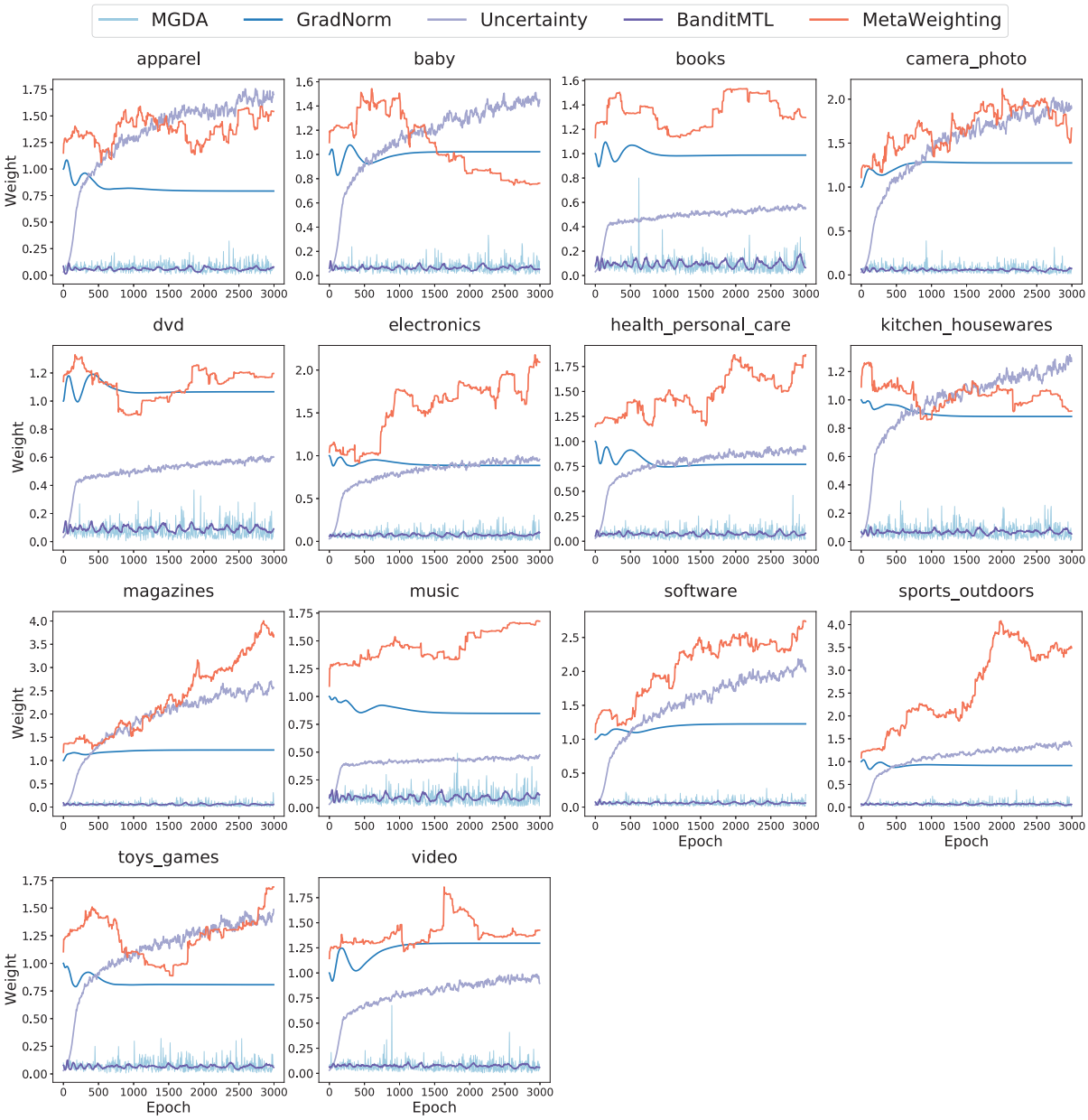


Figure 12: Comparison of task weight adaption processes between MetaWeighting, Uncertainty, Gradnorm, MGDA and BanditMTL for sentiment analysis.