Auxiliary Learning with Joint Task and Data Scheduling

Hong Chen [1] **Xin Wang** [1 2] **Chaoyu Guan** [1] **Yue Liu** [1] **Wenwu Zhu** [1]

Abstract

Existing auxiliary learning approaches only consider the relationships between the target task and the auxiliary tasks, ignoring the fact that data samples within an auxiliary task could contribute differently to the target task, which results in inefficient auxiliary information usage and non-robustness to data noise. In this paper, we propose to learn a joint task and data schedule for auxiliary learning, which captures the importance of different data samples in each auxiliary task to the target task. However, learning such a joint schedule is challenging due to the large number of additional parameters required for the schedule. To tackle the challenge, we propose a joint task and data scheduling (JTDS) model for auxiliary learning. The JTDS model captures the joint task-data importance through a task-data scheduler, which creates a mapping from task, feature and label information to the schedule in a parameter-efficient way. Particularly, we formulate the scheduler and the task learning process as a bi-level optimization problem. In the lower optimization, the task learning model is updated with the scheduled gradient, while in the upper optimization, the task-data scheduler is updated with the implicit gradient. Experimental results show that our JTDS model significantly outperforms the state-of-the-art methods under supervised, semi-supervised and corrupted label settings[1].

1. Introduction

Auxiliary learning, utilizing auxiliary tasks to help improve the target task, has drawn an increasing number of research attentions in the community. This learning paradigm aims to improve the model performance on the target task via utilizing the useful information carried in the related tasks (Navon et al., 2021; Liu et al., 2019a), and has been widely adopted in different areas including image classification (Beyer et al., 2019), recommendation (Wen et al., 2020), vision-language navigation (Zhu et al., 2020) and reinforcement learning (Shelhamer et al., 2017), etc.

A typical and most widely adopted way in auxiliary learning is to calculate the average or total loss of all training samples for each task, linearly combine them into a single loss, and then use the aggregated loss to optimize the task learning model. To achieve promising performance on the target task, the linear weights for task combinations are always tuned with methods such as grid search or Hyper-parameter Optimization (HPO) tools (Kandasamy et al., 2020; Snoek et al., 2012). However, when the number of auxiliary tasks is large, this paradigm generally fails because the complexity of the search space will be exponentially explosive. More recent works (Du et al., 2018; Shi et al., 2020; Lin et al., 2019) utilize the gradient similarity between the target task and the auxiliary tasks to automatically assign weights to different auxiliary tasks in an adaptive way. Furthermore, by using a small set of the target task as guidance, researchers (Navon et al., 2021) propose to learn a nonlinear combination of the given tasks to better utilize the auxiliary information.

Nevertheless, existing works only consider task-level relations, ignoring the important fact that data samples within the same auxiliary task could contribute differently to the target task. Taking the image classification task as an example, when the target task is "correctly classifying a *bird*", "detecting the beak of a bird" becomes a helpful auxiliary task. While an image of a frontal bird face is definitely helpful for the auxiliary task to improve the target task, an image of a bird's back provides little useful information. Moreover, given that real-world data inevitably contain noises, these noisy data samples may counterbalance the benefits of a helpful auxiliary task as well. Therefore, existing works fail to capture the relations of various data samples in different auxiliary tasks towards the target task, suffering from inefficient auxiliary information usage and non-robustness to data noise.

In this paper, we propose to learn a joint task and data

[1]Department of Computer Science and Technology, Tsinghua University [2]THU-Bosch JCML center, Tsinghua University. Correspondence to: Xin Wang <xin_wang@tsinghua.edu.cn>, Wenwu Zhu <wwzhu@tsinghua.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

[1]Our code will be released at https://github.com/forchchch/JTDS

schedule for auxiliary learning, which further captures the importance of different data samples in each auxiliary task to the target task. However, learning such a joint schedule is challenging and still remains largely unexplored. Assuming that the dataset contains m data samples and the number of auxiliary tasks is n, then the number of required parameters for the schedule is $O(mn)$. Considering the data-driven characteristics of current deep learning, even a dataset with 10^4 samples and 1 auxiliary task will result in $2 \cdot 10^4$ parameters. To obtain the optimal value of such large number of parameters, both searching-based and gradient-based methods will be infeasible because of the explosive computational cost, and existing approaches using a small dataset of the target task (Navon et al., 2021) tend to suffer from overfitting problems as indicated in Lorraine et al.'s work (Lorraine et al., 2020). To tackle the challenge, we propose a Joint Task and Data Scheduling (JTDS) model for auxiliary learning. The proposed JTDS model utilizes a novel parameter-efficient task-data scheduler to generate appropriate schedules. Particularly, the task-data scheduler creates a mapping from task, feature and label information to the schedule. This mapping can accurately capture the importance of different samples in each auxiliary task to the target task, while largely reducing the required parameters for the schedule. To jointly optimize the parameters of the task learning model and the task-data scheduler, we formulate the overall learning process as a bi-level optimization problem. In the lower level optimization, we optimize the task learning model parameters under the schedule generated by the task-data scheduler, and in the upper level optimization, we optimize the scheduler parameters using the implicit gradient from a small developing dataset.

We conduct extensive experiments under supervised, semi-supervised and corrupted label settings to validate the effectiveness of our proposed JTDS model. The significant improvement over several state-of-the-art baselines further demonstrates the capability of JTDS in exploiting the auxiliary information and being robust to noisy samples in the data. In summary, we make the following contributions:

- We propose the JTDS model, a novel bi-level optimization based framework for auxiliary learning with joint task and data scheduling.
- We propose a parameter-efficient task-data scheduler, which maps the task, feature and label information to an appropriate schedule. The proposed scheduler is able to capture the importance of different samples in different tasks to the target task while largely reducing the required parameters.
- We conduct extensive experiments to demonstrate the superiority of our proposed JTDS model against several baselines in terms of auxiliary information usage and robustness to data noise.

2. The Proposed Method

In this section, we first formulate our problem, then describe our proposed parameter-efficient task-data scheduler, and finally present the overall learning process of the JTDS model.

2.1. Preliminaries and Problem Formulation

Let $D_t = \{(x_i^t, y_{i1}^t, y_{i2}^t, \cdots, y_{in}^t, y_{iG}^t)\}_{i=1}^m$ be the training set, comprised of m data samples that could be used for $n+1$ different tasks. In the training set, x_i^t is the i^{th} data sample and y_{ik}^t is its label for the k^{th} task $\mathbb{T}_k$. Let $D_v = \{(x_j^v, y_{jG}^v)\}_{j=1}^N$ be the validation set containing N data samples and the corresponding label of each sample under $\mathbb{T}_G$, where G is the index of the target task.

The general auxiliary learning process can be formulated by Eq. (1), where the task learning model $\{f_k\}$ is parametrized by θ and f_k outputs the predictions for task $\mathbb{T}_k$. The training loss function is L_t and the loss optimization process $P()$ generates a set of candidate $\{\theta_p\}$. We choose the one that minimizes the validation error E_v as the parameter for the target task. This practice is commonly adopted in deep learning, where during training, after some pre-defined epochs, we will evaluate our model on the validation set D_v, and finally record the parameter with the best performance as our final model parameter, and T in Eq.(1) is the total evaluation times. Since different f_k generally share some parameters during training, the information of auxiliary tasks can be transferred to the target task. In summary, the goal of auxiliary learning is to design an appropriate L_t to combine the information of the given tasks, so as to obtain the best task learning model (parameterized by θ) for the target task.

$$\theta^* = \arg\min_{\theta_p} E_v(f_G, D_v; \theta_p), \tag{1}$$
$$s.t.\ \{\theta_p\}_{p=1}^T = P(\arg\min_{\theta} L_t(f_1, \cdots, f_n, f_G, D_t; \theta)).$$

One simple but widely adopted method to utilize the auxiliary tasks is to combine the average training loss for each task in a linear way as formulated in Eq. (2), where $l_k(\cdot,\cdot)$ is the loss function for task $\mathbb{T}_k$, and then tune the weight w_k for each task with grid search or other HPO methods. However, these methods suffer from the problem of exponentially explosive searching space when the number of auxiliary tasks is large. Recent approaches propose to assign weights for different tasks based on the gradient similarities between the target task and the auxiliary tasks. Since the gradient similarity is usually changing during the training process, this line of method assigns an adaptive weight to each task. However, these methods are limited in exploiting useful information in auxiliaries due to the ignorance of data

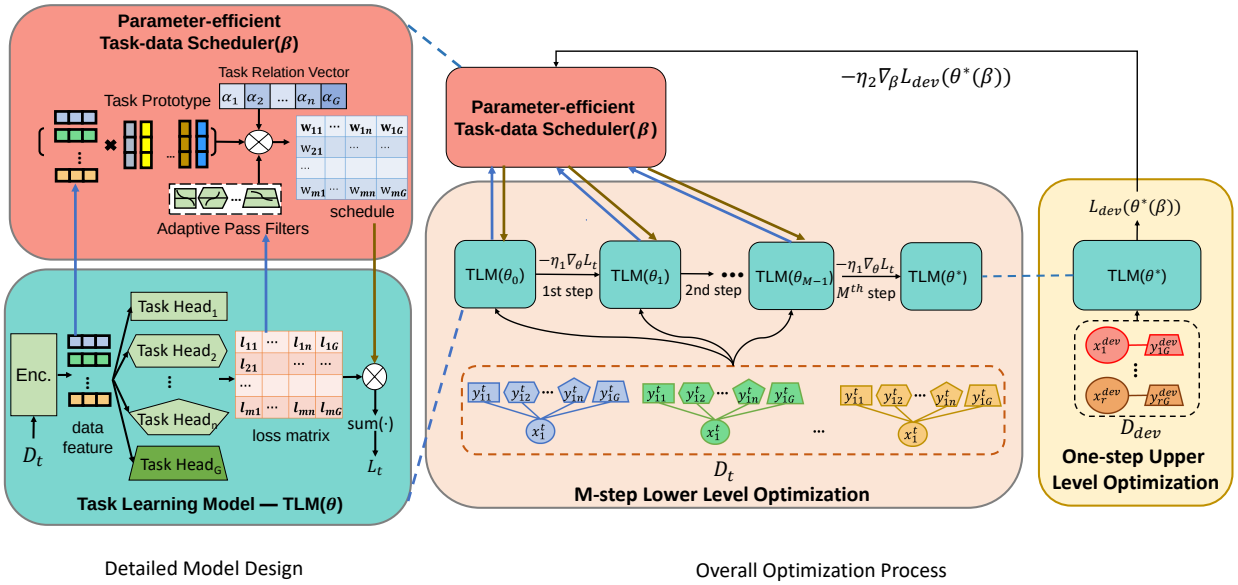


Figure 1. **The overall framework for the JTDS model**. (1)The left part presents the model details of the parameter-efficient task-data scheduler and the task learning model. For the task learning model, we use the most widely used hard parameter sharing architecture(one common backbone encoder and several task-specific heads for each task) as an example, where different tasks are best viewed in shape. (2)The right part illustrates the bi-level optimization process for the task learning model and the task-data scheduler. During the lower-level optimization, the task learning model updates its parameters θ with the scheduled gradient for M times on D_t. After the M-step lower level optimization, the loss of the task learning model on D_{dev} is calculated to obtain the implicit gradient $\nabla_\beta L_{dev}$, which we use to update β, the parameters of the scheduler.

sample level information.

$$L_t = \sum_{k\in\{1,\cdots,n,G\}} w_k \cdot \frac{1}{m}\sum_{i=1}^{m} l_k(f_k(x_i^t), y_{ik}^t; \theta). \quad (2)$$

Our Solution To tackle this problem, we propose to jointly schedule task and data sample for auxiliary learning. In particular, we provide the task learning model with a schedule about the importance of each data sample within each task to the target task. By denoting all the training task ID as $U = \{1, 2, \cdots, n, G\}$, different from Eq. (2), our training optimization objective L_t is expressed in Eq. (3), where w_{ik} gives the importance of sample i within task $\mathbb{T}_k$. The key challenge is how to decide the value of the totally $m(n+1)$ parameters $\{w_{ik}\}$. As aforementioned, to directly optimize such a large number of parameters is challenging, thus designing a parameter-efficient scheduler to generate w_{ik} is necessary. To this end, we will in detail describe our parameter-efficient task-data scheduler in the next subsection.

$$L_t = \sum_{i=1}^{m}\sum_{k\in U} w_{ik} \cdot l_k(f_k(x_i^t), y_{ik}^t; \theta). \quad (3)$$

For easier understanding towards the proposed method, we list the following used notations in Table 1.

Table 1. Notations

Notation	Description
θ	the parameters of the task learning model
$\boldsymbol{\alpha}$	the task relation vector
$f_{k,enc}$	the encoder for the k^{th} task
$\boldsymbol{P}_k$	the prototype vector of the k^{th} task
a_k, b_k	the parameters of the k^{th} adaptive pass filter
β	the whole parameters of the task-data scheduler
K	the number of truncated Neumann series terms
M	the lower optimization steps
$\boldsymbol{c}_{ik}$	the feature of the i^{th} sample within task $\mathbb{T}_k$
$interval$	the interval between two evaluations on D_v
D_{dev}, L_{dev}	the developing dataset and the target loss on D_{dev}

2.2. The Parameter-efficient Task-data Scheduler

Given all the training data samples and tasks, the scheduler outputs $\{w_{ik}\}$, the schedule that captures the importance of each data sample within each task to the target task. The Detailed Model Design part of Figure 1 demonstrates details of our proposed Parameter-efficient Task-data Scheduler and its interactions with the task learning model (TLM).

To decide an appropriate schedule while using fewer parameters, a natural practice is to relate the schedule to existing information. With consideration to the factors that influence the importance of each data sample within each task to the

target task, we propose two hypotheses that can help to build the parameter-efficient task-data scheduler. The first hypothesis focuses on the relationships between the target task and each data sample within each auxiliary task:

Hypothesis 1. Data sample x_i^t in auxiliary task $\mathbb{T}_k$ is beneficial to the target task $\mathbb{T}_G$, if task $\mathbb{T}_k$ is beneficial to the target task $\mathbb{T}_G$ and the pair (x_i^t, y_{ik}^t) is beneficial to task $\mathbb{T}_k$.

The first hypothesis indicates the task and the data sample make multiplicative contribution to the target task. Either $\mathbb{T}_k$ has low relevance to $\mathbb{T}_G$ or (x_i^t, y_{ik}^t) is less informative for $\mathbb{T}_k$ will make $l_k(f_k(x_i^t), y_i^t; \theta)$ carry little importance. This hypothesis enables that i) the same data sample within different auxiliary tasks can express different levels of importance in terms of the relevance to the target task, and ii) different data samples in the same auxiliary task $\mathbb{T}_k$ can also demonstrate different levels of importance based on how informative the data sample is to $\mathbb{T}_k$.

The second hypothesis, on the other hand, focuses on evaluating how beneficial the training pair (x_i^t, y_{ik}^t) is to $\mathbb{T}_k$:

Hypothesis 2. (x_i^t, y_{ik}^t) is beneficial to task $\mathbb{T}_k$, if x_i^t contains useful features for $\mathbb{T}_k$ and y_{ik}^t is a correct label.

The second hypothesis suggests that the feature of x_i^t and the loss term $l_k(f_k(x_i^t), y_i^t; \theta)$ can provide a hint on the importance of (x_i^t, y_{ik}^t) to $\mathbb{T}_k$. Here we note that the correctness of a label can be determined by its corresponding loss value (Arazo et al., 2019).

The first hypothesis decomposes w_{ik} into two parts, i.e., (1) the relevance between $\mathbb{T}_k$ and $\mathbb{T}_G$, and (2) the importance of (x_i^t, y_{ik}^t) to $\mathbb{T}_k$. The second hypothesis further decomposes part (2) into the importance of the sample feature and the rationality of the label. Based on the two hypotheses, we propose the parameter-efficient task-data scheduler.

We first introduce a learnable task relation vector $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_n, \alpha_G]$ to describe the relationships between each task $\mathbb{T}_k$ and the target task $\mathbb{T}_G$, where each α_k represents the importance of $\mathbb{T}_k$ to $\mathbb{T}_G$. Additionally, the importance of (x_i^t, y_{ik}^t) to $\mathbb{T}_G$ relies on the sample feature and sample loss. Within each task $\mathbb{T}_k$, the data sample x_i^t will be sent to the task learning model f_k to obtain its feature $\mathbf{c}_{ik} = f_{k,enc}(x_i^t)$ and its loss $l_k(f_k(x_i^t), y_{ik}^t; \theta)$, where $f_{k,enc}(\cdot)$ is the encoder of the task learning model for $\mathbb{T}_k$. To judge the importance of (x_i^t, y_{ik}^t) to $\mathbb{T}_k$ from the feature perspective, we introduce a set of learnable task prototypes $\mathbb{P} = \{\boldsymbol{P}_k\}_{k \in U}$. Each $\boldsymbol{P}_k$ in $\mathbb{P}$ is a vector that has the same dimension with $\mathbf{c}_{ik}$, representing the feature prototype of task $\mathbb{T}_k$. These prototypes are used to evaluate whether data sample x_i^t contains useful features for $\mathbb{T}_k$, preventing feature level noise from x_i^t. Inspired by the idea that the normalized loss distribution of clean and noisy data can be modeled with two beta distributions parameterized by two groups of parameters (Arazo et al., 2019), the scheduler assigns each task with an adaptive pass filter to filter out the noisy pairs. These filters are flexible to different scenarios by introducing learnable parameters $\{a_k\}_{k \in U}$ and $\{b_k\}_{k \in U}$. By denoting $l_k(f_k(x_i^t), y_{ik}^t; \theta)$ as l_{ik}, the scheduler generates the joint schedule w_{ik} as shown in Eq. (4), where $\sigma()$ is the activation function to ensure non-negative values. The second term utilizes the inner product between the task prototype and the sample feature to evaluate the informativeness of x_i^t. The third term is a linear classifier to judge whether (x_i^t, y_{ik}^t) is a noisy pair for $\mathbb{T}_k$, and $\hat{l}_{ik}$ is the loss after normalization. Considering that w_{ik} relies on the loss, feature and the introduced learnable parameters, which are always updated during the training process, w_{ik} is naturally adaptive.

$$w_{ik} = \sigma(\alpha_k) \cdot \sigma(\ \boldsymbol{P}_k^T \boldsymbol{c}_{ik}\ ) \cdot \sigma(a_k \hat{l}_{ik} + b_k). \tag{4}$$

The proposed scheduler comprehensively considers task, feature and label information, while only introducing an additional learnable parameter set $\beta = \{\boldsymbol{\alpha}, \mathbb{P}, \{a_k, b_k\}_{k \in U}\}$, where the number of parameters is $O(dn)$ and d is the dimension of the features if we assume that the features of all tasks have the same dimension d. Considering the fact that the feature dimension d (typical value $\{32, 64, 128, 256\}$) is usually much smaller than the dataset size m, the parameters to optimize are therefore largely reduced. In the next subsection, we present the overall auxiliary learning process and optimization for parameters θ and β.

2.3. JTDS Overall Framework and Learning Algorithm

The overall framework of the JTDS model is presented in Figure 1, containing two main components, i.e., the parameter-efficient task-data scheduler and the task learning model for general auxiliary learning. We expect the task-data scheduler to provide an appropriate joint schedule for training the task learning model, so that we could obtain a task learning model that has excellent performance on the target task $\mathbb{T}_G$. The task learning model parameters θ are updated with the goal of minimizing the loss in Equation (3). To optimize the parameters β in the scheduler, we introduce another small developing dataset $D_{dev} = \{(x_i^{dev}, y_{iG}^{dev})\}_{i=1}^r$. The dataset D_{dev} only contains information about task $\mathbb{T}_G$, i.e., the target task, which is a small subset sampled from the validation set D_v. Given that the goal for the scheduler is to obtain an optimal training schedule for the target task on D_v, the loss on D_{dev} can be utilized to update the parameters β. Formally, our problem can be formulated as a bi-level optimization problem shown in Eq. (5), where $L_{dev}(\theta^*(\beta)) = \sum_{i=1}^r l_G(f_G(x_i^{dev}), y_{iG}^{dev}; \theta^*(\beta))$, and $L_t(\theta, \beta)$ is the scheduled training loss in Eq. (3). Note that $w_i k$ is parameterized by β.

$$\begin{aligned} \beta^* &= \arg\min_{\beta} L_{dev}(\theta^*(\beta)), \\ s.t.\ \theta^* &= \arg\min_{\theta} L_t(\theta, \beta). \end{aligned} \tag{5}$$

During the lower level optimization, with the parameters of the scheduler β fixed, we update θ, the parameters of task learning model. θ is updated by using the scheduled gradient (weighted gradient sum of the data samples within different tasks) as shown in Eq. (6).

$$\nabla_\theta L_t(\theta, \beta) = \sum_{k \in U} \sum_{i=1}^{m} w_{ik} \nabla_\theta l_k(f_k(x_i^t), y_{ik}^t; \theta). \quad (6)$$

During the upper level optimization, we need to derive the gradient of $L_{dev}(\theta^*(\beta))$ with respect to β. Given that $L_{dev}(\theta^*(\beta))$ directly relies on θ instead of β, we follow the literature (Lorraine et al., 2020) and utilize implicit differentiation to obtain this implicit gradient. With Theorem 2.1, we can obtain the gradient of $L_{dev}(\theta^*(\beta))$ with respect to β using the chain rule as in shown in Eq. (7). For detailed derivation of the implicit gradient, see Appendix A.

$$\begin{aligned} &\nabla_\beta L_{dev}(\theta^*(\beta)) = \nabla_\theta L_{dev} \cdot \nabla_\beta \theta^* \\ &= -\nabla_\theta L_{dev} \cdot (\nabla_\theta^2 L_t)^{-1} \cdot \nabla_\beta \nabla_\theta L_t|_{(\beta, \theta^*(\beta))}. \end{aligned} \quad (7)$$

Theorem 2.1. *(Cauchy,Implicit Function Theorem). If there exists one point (θ_0, β_0) where $\nabla_\theta L_t(\theta, \beta) = 0$ and the regularity conditions are satisfied, then within the neighborhood of (θ_0, β_0), there exists a implicit function $\theta^*(\beta)$ s.t. $\nabla_\theta L_t(\theta, \beta) = 0|_{\beta, \theta^*(\beta)}$ and the derivative of θ^* w.r.t. β is: $\nabla_\beta \theta^* = -(\nabla_\theta^2 L_t)^{-1} \cdot \nabla_\beta \nabla_\theta L_t|_{(\beta, \theta^*(\beta))}$.*

However, directly computing the inverse of the Hessian is intractable for deep models. We adopt the K-truncated Neumann series to approximate this inverse as illustrated in Eq. (8). By using this approximation to approach the inverse of the Hessian, the implicit gradient $\nabla_\beta L_{dev}(\theta^*(\beta))$ can be calculated in Eq. (9).

$$(\nabla_\theta^2 L_t)^{-1} = \sum_{i=0}^{\infty} (I - \nabla_\theta^2 L_t)^i \approx \sum_{i=0}^{K} (I - \nabla_\theta^2 L_t)^i, \quad (8)$$

$$\nabla_\beta L_{dev} = -\nabla_\theta L_{dev} \cdot \sum_{i=0}^{K} (I - \nabla_\theta^2 L_t)^i \cdot \nabla_\beta \nabla_\theta L_t. \quad (9)$$

Upon obtaining the gradient of θ and β, Algorithm 1 presents the complete algorithm that simultaneously learns the task learning model and the task-data scheduler. During the lower level optimization, the parameters β are fixed, and the scheduler $Scheduler(;\beta)$ gives the importance w_{ik} to each sample within each task as in Eq. (4). The parameters θ are updated with gradient in Eq. (6) at the learning rate η_1. Instead of waiting θ to converge, we conduct the more efficient M-step optimization as in (Lorraine et al., 2020), i.e., after θ has been updated for M times, we switch to the upper optimization to optimize β. According to Eq. (9), we first calculate L_{dev} and L_t, and then use the implicit gradient to update β at the learning rate η_2. To avoid the task learning model overfitting to D_{dev}, after each $interval$ iterative upper and lower optimization, we test the task learning model on the validation set D_v and record the θ with the lowest error as our final learned model for the target task as in Eq. (1).

Discussion w_{ik} is also a function of θ because both $f_{k,enc}(x_i^t)$ and l_{ik} rely on θ as shown in Eq. (4), but when we update θ in Eq. (6), we only consider the gradient of l_{ik} w.r.t. θ and ignore the gradient of w_{ik} to θ because our target is to find the most useful task-data pairs and use their loss to train the task learning model. In fact, if we consider the gradient of w_{ik} w.r.t. θ, it is likely that we obtain some trivial solution of θ. One possible solution could be that θ minimizes L_t by forcing all w_{ik} equal to zero. In such situation, the model will ignore the loss of each task and data sample pair, thus failing to learn any knowledge about the tasks in the lower optimization. Therefore, the gradient in Eq. (6) only involving the loss l_{ik} is reasonable. For consistency, during the upper optimization, we drop the gradient of the feature $\boldsymbol{c}_{ik}$ and loss $\hat{l}_{ik}$ w.r.t. θ using the *detach()* operation. Additionally, although in this paper, we only consider the scenario that all the tasks share the same data samples with different labels, our method can also be applied to the scenario where each task has an independent dataset. The joint task-data consideration can make it handle some domain adaptation (Wang & Deng, 2018) or multimodal scenarios (Zhu et al., 2015; Wang et al., 2021a) with both task and data level adaptation.

3. Experiments

In this section, we empirically assess the efficacy of our proposed JTDS in exploiting the auxiliary information and whether the parameter-efficient scheduler provides a reasonable task and data schedule.

3.1. Experimental Setup

Task and Dataset We first evaluate our methods on the fine-grained bird classification problem on the CUB (Wah et al., 2011) dataset, which contains 11788 images of 200 bird species. There is an associated set to each image that describes its 312 visual attributes, e.g., head color. In general, it is difficult for non-domain-experts to classify these birds, but it is much easier to discriminate the visual attributes. Therefore, a natural practice is to utilize the attribute classification as auxiliary tasks to help the bird species classification. Thus, we regard the bird classification problem as the target task and the 312 attribute classification problem as the auxiliary tasks. There are total 313 tasks for training. Then, we focus on one currently popular practice that utilizes the self-supervised learning task as auxiliary tasks (Beyer et al., 2019; Mangla et al., 2020), where the target task is the general image classification problem and the rotation degree

Algorithm 1 JTDS for Auxiliary Learning

Input: D_t, D_{dev}, D_v, $interval$,η_1, η_2, K, M;
Initialize θ, β, θ_{opt}, $t = 0$, $error_v = 1.0$;
while not converged **do**
  // lower optimization
  for $j = 1$ **to** M **do**
    for $(x_i^t, y_{ik}^t) \in D_t$ **do**
      $l_{ik} = l_k(f_k(x_i^t), y_{ik}^t; \theta)$, $\hat{l}_{ik} = norm(l_{ik})$,
      $\boldsymbol{c}_{ik} = f_{k,enc}(x_i^t)$, $w_{ik} = Scheduler(\hat{l}_{ik}, \boldsymbol{c}_{ik}; \beta)$;
    end for
    $\theta \leftarrow \theta - \eta_1 \sum_{k \in U} \sum_{i=1}^{m} w_{ik} \nabla_\theta l_{ik}$;
  end for
  $t \leftarrow t + 1$;
  // record the best parameters on D_v
  if $t\%interval == 0$ **then**
    $error = E_v(f_G, D_v; \theta)$;
    if $error < error_v$ **then**
      $error_v = error, \theta_{opt} = \theta$;
    end if
  end if
  // upper optimization
  $L_{dev} = \sum_{i=1}^{r} l_G(f_G(x_i^{dev}), y_{iG}^{dev}; \theta)$;
  for $(x_i^t, y_{ik}^t) \in D_t$ **do**
    $\hat{l}_{ik} = norm(l_k(f_k(x_i^t), y_{ik}^t; \theta)).detach()$,
    $\boldsymbol{c}_{ik} = f_{k,enc}(x_i^t).detach()$,
    $w_{ik} = Scheduler(\hat{l}_{ik}, \boldsymbol{c}_{ik}; \beta)$;
  end for
  $L_t = \sum_{k \in U} \sum_{i=1}^{m} w_{ik} * l_{ik}$;
  $\beta \leftarrow \beta + \eta_2 \nabla_\theta L_{dev} \cdot \sum_{i=0}^{K} (I - \nabla_\theta^2 L_t)^i \cdot \nabla_\beta \nabla_\theta L_t$;
end while
Return θ_{opt}

prediction problem is the auxiliary task. We conduct this experiment on CIFAR10, CIFAR100 (Krizhevsky & Hinton, 2009) and Oxford-IIIT Pet dataset (Parkhi et al., 2012), and in the auxiliary task, we rotate each image with degrees of $\{0, 90, 180, 270\}$. CIFAR10 is a widely used image classification dataset that contains 60000 images with 10 categories and CIFAR100 with 100 categories. and Oxford-IIIT Pet contains 7349 images of 37 species of pets. Except for image classification, we also evaluate our method on the recommendation task on the MovieLens-1M dataset (Harper & Konstan, 2016), where we regard the rating prediction as the target task and the CTR prediction as the auxiliary task. The input for the recommendation task is the features of the users and items provided in the dataset. The MovieLens-1M dataset contains 1 million user-item interaction.

Baselines We compare our methods with several auxiliary learning methods and dynamic weighting multitask learning methods. Single task learning(STL) is a proper baseline to judge whether the auxiliary information benefits the target task. Another natural baseline is the naive auxiliary learning method(NAL) that combines the main loss and the auxiliary loss in a linear way and tune the weights with some search algorithms as conducted in (Beyer et al., 2019). Gradient cosine similarity(GCS) (Du et al., 2018) is an adaptive weighting methods for auxiliary learning by using the gradient similarity between the main task and the auxiliary tasks. AuxL (Navon et al., 2021) is a recent method that proposes to dynamically learn non-linear combinations of different tasks to better utilize the auxiliary information. Uncertainty (Kendall et al., 2018) is a multitask learning dynamic weighting methods that weight each task according to its uncertainty. N-JTDS is a naive implementation of our proposed joint task and data schedule where we directly assign a learnable weight to each sample within each task, and also optimize these $m(n+1)$ weights with the same bi-level optimization process. Detailed implementation algorithm of the N-JTDS can be found in the Appendix A.

Implementation Details In our experiments, we adopt the most widely used hard parameter sharing structure for the task learning model(one backbone followed by several task heads). In the CUB, Pet, CIFAR100 dataset, we use the ResNet18 (He et al., 2016) as the backbone, which is used as the aforementioned encoder for the task learning model, i.e., $f_{k,enc}$, and all the tasks share the same backbone encoder. Particularly, to fit the smaller image size in CIFAR100, we replace the original 7x7 convolution kernel to 3x3 and remove the maxpooling layer in the first block. In CIFAR10, we use a 4-layer ConvNet as the backbone, and in MovieLens-1M, we use the AutoINT (Song et al., 2019) as the backbone. As for the task heads, we adopt the Multi-layer Perceptron(MLP) with number of layers searched in {1,2} by the STL baseline. The datasets are split into training, validation and test set as the officially recommended. We sample 200 samples from the validation set to form D_{dev}. For the baselines that do not use D_{dev} as guidance, we add these samples to their training set for fair comparison. For more implementation details, please refer to Appendix A.

3.2. Auxiliary Information Usage

To assess whether our proposed method has stronger ability in exploiting auxiliary information, we conduct experiments on two different settings, the fully supervised setting and semi-supervised setting for the target task. For the fully supervised setting, all training samples are equipped with labels on the target task. However, on the semi-supervised setting(with the CUB, Pet, and CIFAR100 dataset), only 20% of the training samples have labels on the target task. Considering that NAL and GCS baseline cannot handle such large number of auxiliary tasks in CUB dataset, we regard all loss of the auxiliary tasks as a whole to implement these two methods in CUB dataset. The image classification task is evaluated with the accuracy and the rating prediction is evaluated with RMSE(rooted mean square error) as the metric. The experimental results are presented in Table 2 and Table 3 where we run three random seeds for each experiment and report the mean performance.

In the semi-supervised setting, all the methods that utilize the auxiliary tasks outperforms the STL baseline, indicating that these auxiliary tasks are quite beneficial when the labels for the target task are scarce. Our method, JTDS, achieves the best performance among all the auxiliary methods, showing its ability in better auxiliary information usage. In the supervised setting, JTDS also outperforms all the baselines on the 5 datasets. Particularly, in the CUB, CIFAR10 and CIFAR100 experiments, most auxiliary methods fail to outperform the STL baseline, indicating that the auxiliary tasks on average are not so beneficial when the labels for the target task are adequate. This phenomenon has also been pointed out by (Navon et al., 2021). However, our method still surprisingly brings substantial improvement over STL, showing the power of the joint task and data schedule. This improvement to some extent fits our intuition that the beneficial and harmful information under the same auxiliary task will counterbalance each other, where only considering task level relations cannot make full use of the auxiliary information. Additionally, the results of the N-JTDS and JTDS show that directly optimizing $m(n+1)$ parameters will suffer different degree of overfitting problem. In the CIFAR10, CIFAR100 and the Pet experiments where the number of tasks is 2, the data sample number is comparatively small, and the number of required schedule parameters for N-JTDS is not so large, N-JTDS shows comparatively acceptable ability in using auxiliary information. However, in the CUB experiment, when the number of auxiliary tasks is 312 and the number of the required schedule parameters for N-JTDS reaches million level, N-JTDS suffers large performance drop as indicated in (Lorraine et al., 2020), which is similar in ML-1M. In contrast, our method, JTDS, shows consistent superiority over all the baselines. Overall, our proposed JTDS model has better ability in utilizing the auxiliary information under different settings and shows excellent scalability to number of samples and tasks.

Table 2. Performance of different methods under the fully-supervised setting(CF-10, CF-100, ML-1M respectively represents the CIFAR10, CIFAR100, MovieLens-1M dataset).

Metric	Accuracy(%)				RMSE
Method	**CUB**	**Pet**	**CF-10**	**CF-100**	**ML-1M**
STL	73.86	61.45	71.60	74.14	0.9112
NAL	73.42	66.09	70.42	73.38	0.9101
Uncertainty	72.54	67.14	70.93	68.10	0.9103
GCS	73.70	66.30	70.64	74.14	0.9098
AuxL	74.32	66.30	71.23	73.80	0.9097
N-JTDS	71.38	67.28	70.57	75.06	0.9181
JTDS(ours)	**77.04**	**70.01**	**72.59**	**75.68**	**0.9087**

3.3. Robustness to Corrupted Labels

To further assess the ability of our model in distinguishing beneficial and harmful information, we conduct experiments under a more severe setting where the training labels for the target task are corrupted. On the CUB dataset, the labels for the bird species are uniformly corrupted by different ratios as in (Zhang et al., 2017). The results are shown in Figure 2.

Table 3. Image classification accuracy(%) of different methods under the semi-supervised setting.

Method	**CUB**	**Pet**	**CIFAR100**
STL	38.35	30.21	55.16
NAL	48.15	38.00	57.52
Uncertainty	46.66	43.57	57.70
GCS	46.50	36.80	55.66
AuxL	50.19	36.42	55.94
N-JTDS	46.50	45.53	57.54
JTDS	**51.21**	**53.49**	**58.56**

Comparing the results of STL with that of the auxiliary methods, we find that auxiliary information plays a more and more important role as the corrupted ratio becomes larger. This phenomenon is consistent to the semi-supervised setting where the useful labels of the target task are scarce. Among all the methods using auxiliary tasks, it is obvious that our proposed method shows best robustness to corrupted labels, which benefits from joint consideration to the task and data sample schedule.

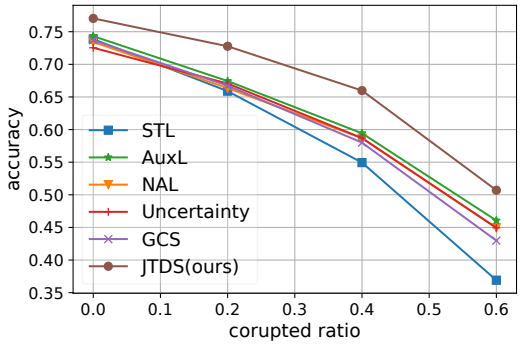
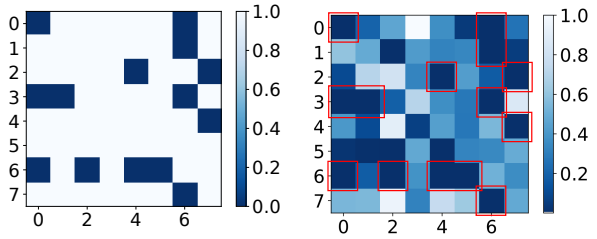


Figure 2. Accuracy of different models under different ratios of corrupted labels

To further show the ability of our method in detecting the harmful information, we randomly sample 64 training images within the target task in the 0.2 corrupted setting, and present the 64 scheduled weights in Figure 3(b). In Figure 3(a), we present the ground truth whether the sampled image label is correct, where 0.0 means corrupted label while 1.0 means correct label. Comparing the dark color part of the two figures, we find that the corrupted ones under the target task are all scheduled with extremely small weights. This result shows that our proposed method can alleviate the impact of the harmful information by decreasing

the weights for the harmful samples. The schedule provided by our scheduler is reasonable and fits human intuition.



(a) label correctness (b) scheduled weights

Figure 3. Corrupted Sample Detection

3.4. Effectiveness Analysis

To further validate the impact of different components proposed in the scheduler, we conduct ablations about variants of our proposed method on the CUB dataset under all the aforementioned settings. The results are shown in Table 4. We consider four variants of our methods, i.e., 1)only task: the scheduler only assigns each task with a common weight for all samples within this task, 2)w/o loss: remove the term about normalized loss in Equation (4), 3)w/o feature: remove the similarity between each sample feature and the task prototype in Equation (4), 4)w/o aux, we only train on the target task while ignoring the auxiliary tasks, where our method degenerates to scheduling data for the target task.

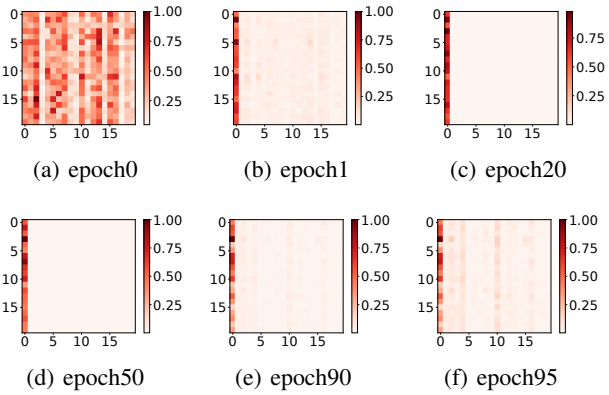
The comparison between only task and our method again shows the importance of joint considering the task and data level information. The results of w/o loss shows that the loss information plays an important role for the scheduler to detect the noisy samples. Without the loss information, performance on the corrupted settings largely drops. Also, feature is an important factor to decide the importance of a data sample within a task. Without using the feature, the method shows poor auxiliary information usage under the full and semi-supervised setting. These results fit our hypothesis that both feature and label are necessary for deciding whether a sample is suitable for a task. The w/o aux results again show that when the useful information for the target task becomes less, auxiliary tasks become more important. Interestingly, the results of STL and w/o aux show that only data schedule can also help to improve the model performance to some extent. This improvement could be explained by the benefits of curriculum learning (Wang et al., 2021b), where different methods reweight the data according to different metrics so as to obtain better model performance. From this perspective, under one task scenario, our method could serve as an automatic curriculum learning that simultaneously utilizes the loss and feature information to reweight the data. The results also show that this variant of our method could give a promising curriculum.

Table 4. CUB Fine-grained image classification accuracy (%).

	settings				
			corrupted ratio		
Method	full	semi	0.2	0.4	0.6
STL	73.86	38.35	65.89	54.94	36.91
only task	72.90	39.97	65.93	54.82	40.95
w/o loss	76.33	49.23	67.24	57.82	44.24
w/o feature	73.97	44.63	67.14	59.79	47.15
w/o aux	75.47	38.44	70.18	61.84	39.65
JTDS	**77.04**	**51.21**	**72.77**	**65.98**	**50.70**

3.5. Learning Behavior

Joint task-data schedule We visualize how the schedule changes during the training process to find some training pattern. Under the supervised setting on the CUB dataset, we present the schedule for 20 samples within 20 tasks in Figure 4, where each column represents weights of the 20 samples within one task and task 0 is the target task, i.e., fine-grained image classification. The results show that the schedule will quickly focus on the target task within a few steps and then the model will keep training on the target task for a period. This indicates that our method has the ability to discover the most beneficial information. In the later training process, the model will also gradually increase the weights of other auxiliary tasks while keeping on the target task. This behavior could be explained as to prevent overfitting on the target training set, and the loss of the auxiliary tasks will serve as a regularization term. Additionally, the weights of the samples under the same task are different from each other, indicating that it is necessary to jointly taking the sample and task level information into consideration. For more learning behavior of the models, please refer to the Appendix A.



(a) epoch0 (b) epoch1 (c) epoch20

(d) epoch50 (e) epoch90 (f) epoch95

Figure 4. Schedule Evolving

4. Related Work

Auxiliary Learning Auxiliary learning aims at improving the model performance on the task of interest by utilizing the information of related auxiliary tasks. Most of existing works use auxiliary tasks by linearly combining the auxiliary losses and the main loss, and then use search methods like grid search to tune the linear weights (Beyer et al., 2019; Wen et al., 2020). However, this simple and widely used method generally fails when the number of auxiliary tasks is large because the search space complexity is exponential explosive. To find the beneficial auxiliary tasks, some recent works (Lin et al., 2019; Du et al., 2018; Shi et al., 2020) utilize the parameter gradient similarity between the main task and the auxiliary tasks to weight each task loss. Furthermore, Navon et al. propose to learn nonlinear combinations of different tasks to better utilize the auxiliary information.

However, these existing works all average the data level effect to consider the relationships among tasks, causing counterbalance between the useful and useless data information within a task and leading to sub-optimal information usage. This may be the reason why in (Navon et al., 2021) the existing auxiliary methods fail to achieve better performance than single task learning with full supervision.

Multitask Learning Another line of highly related works to auxiliary learning is multitask learning (Crawshaw, 2020), where there are several equally important tasks to learn. Multi-task learning aims to obtain a model that performs well on all the given tasks, different from auxiliary learning that only concerns the performance on the main task. Existing multitask learning methods mainly contain the multitask architecture design, the multitask learning optimization, and the multitask relationship learning. The multitask architecture design aims to design proper architectures to decide which parameters to share under different tasks (Liu et al., 2019b; Gao et al., 2019; Qin et al., 2021), the multitask optimization focus on optimizing the architecture parameters by using methods like loss weighting (Kendall et al., 2018; Chen et al., 2018), and the multitask relationship learning focuses on obtaining the explicit relationships among tasks (rel; Dwivedi & Roig, 2019; Zhao et al., 2019).

Curriculum Learning Curriculum learning points out that the order of presenting data will influence the model performance. The curriculum learning aims to define a proper data training order so that the model can achieve faster convergence and better performance (Wang et al., 2021b). Conventional curriculum learning works follow the easy-to-hard training paradigm (Platanios et al.; Spitkovsky et al., 2010; Wei et al., 2017; Chen et al., 2021b). However, these methods generally need domain-specific knowledge to define the hardness of samples, hard to be applied to the general settings. Later works rely on the loss of each sample to judge the sample hardness (Kumar et al.; Meng et al., 2017), named self-paced learning. The model will start learning from the samples with small loss in self-paced learning. However, this easy-to-hard paradigm does not always work well. To tackle the problem, more recent works (Saxena et al., 2019; Raghu et al., 2021; Wang et al., 2020; Chen et al., 2021a; Zhang et al., 2022) propose to learn the curriculum in an automatic way. However, these works only focus on learning with a single task, failing to consider the influence of a sample within an auxiliary task to the target task. Specifically, DDS (Wang et al., 2020) also adopts a bi-level optimization framework to select data, but it cannot handle the joint task-data selection for auxiliary learning because the required parameters for the selector will be extremely large, which is both memory-consuming and hard to optimize with a small developing dataset.

Position of Our Work Our work focuses on the auxiliary learning where we utilize the information of auxiliary tasks to help of the task of interest. The further fine-grained consideration to data samples is inspired by curriculum learning (Wang et al., 2021b). Our method can be regarded as a special kind of curriculum learning that simultaneously schedules the task and data samples, which is one further step towards the self-directed machine learning (Zhu et al., 2022). Additionally, in this work, we explicitly investigate the factors that could influence the schedule, including task, sample feature and sample label, and the experiments validates the effectiveness of these factors, which is scarcely explored before. Further works can explore more ways to combine these factors or investigate other factors that will influence the schedule. The limitation of this work is that it requires a small developing dataset that contains clean data of the target task, how to obtain a proper schedule without the clean dataset is also an interesting future direction.

5. Conclusion

In this paper, we propose to give a joint task and data schedule for auxiliary learning. The proposed parameter-efficient task-data scheduler offers an appropriate schedule while using much fewer additional parameters. To optimize the task learning model and the scheduler, we design a bi-level optimization based algorithm. Our method shows better ability in auxiliary information usage under supervised, semi-supervised and corrupted label settings, promising to many applications in real life scenario.

Acknowledgement

This work is supported by the National Key Research and Development Program of China No. 2020AAA0106300National Natural Science Foundation of China No. 62102222 and partially funded by THU-Bosch JCML center.

References

Arazo, E., Ortego, D., Albert, P., O'Connor, N., and McGuinness, K. Unsupervised label noise modeling and loss correction. In *International Conference on Machine Learning*, pp. 312–321, 2019.

Beyer, L., Zhai, X., Oliver, A., and Kolesnikov, A. S4L: self-supervised semi-supervised learning. In *ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 1476–1485, 2019.

Chen, H., Chen, Y., Wang, X., Xie, R., Wang, R., Xia, F., and Zhu, W. Curriculum disentangled recommendation with noisy multi-feedback. *Advances in Neural Information Processing Systems*, 34, 2021a.

Chen, Y., Wang, X., Fan, M., Huang, J., Yang, S., and Zhu, W. Curriculum meta-learning for next poi recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2692–2702, 2021b.

Chen, Z., Badrinarayanan, V., Lee, C., and Rabinovich, A. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML 2018*, pp. 793–802. PMLR, 2018.

Crawshaw, M. Multi-task learning with deep neural networks: A survey. *CoRR*, 2020.

Du, Y., Czarnecki, W. M., Jayakumar, S. M., Pascanu, R., and Lakshminarayanan, B. Adapting auxiliary losses using gradient similarity. *CoRR*, 2018.

Dwivedi, K. and Roig, G. Representation similarity analysis for efficient task taxonomy & transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 12387–12396. Computer Vision Foundation / IEEE, 2019.

Gao, Y., Ma, J., Zhao, M., Liu, W., and Yuille, A. L. NDDR-CNN: layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 3205–3214, 2019.

Harper, F. M. and Konstan, J. A. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5 (4):19:1–19:19, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, 2016.

Kandasamy, K., Vysyaraju, K. R., Neiswanger, W., Paria, B., Collins, C. R., Schneider, J., Póczos, B., and Xing, E. P. Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly. *J. Mach. Learn. Res.*, pp. 81:1–81:27, 2020.

Kendall, A., Gal, Y., and Cipolla, R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 7482–7491, 2018.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *Computer Science*, 2014.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. *Handbook of Systemic Autoimmune Diseases*, 1(4), 2009.

Kumar, M. P., Packer, B., and Koller, D. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, pp. 1189–1197.

Lin, X., Baweja, H. S., Kantor, G., and Held, D. Adaptive auxiliary task weighting for reinforcement learning. In *NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 4773–4784, 2019.

Liu, S., Davison, A. J., and Johns, E. Self-supervised generalisation with meta auxiliary learning. In *NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 1677–1687, 2019a.

Liu, S., Johns, E., and Davison, A. J. End-to-end multi-task learning with attention. In *CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 1871–1880, 2019b.

Lorraine, J., Vicol, P., and Duvenaud, D. Optimizing millions of hyperparameters by implicit differentiation. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, Proceedings of Machine Learning Research, pp. 1540–1552. PMLR, 2020.

Mangla, P., Singh, M., Sinha, A., Kumari, N., Balasubramanian, V. N., and Krishnamurthy, B. Charting the right manifold: Manifold mixup for few-shot learning. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2020, Snowmass Village, CO, USA, March 1-5, 2020*, pp. 2207–2216. IEEE, 2020.

Meng, D., Zhao, Q., and Jiang, L. A theoretical understanding of self-paced learning. *Inf. Sci.*, 414:319–328, 2017.

Navon, A., Achituve, I., Maron, H., Chechik, G., and Fetaya, E. Auxiliary learning by implicit differentiation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. V. Cats and dogs. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3498–3505, 2012.

Platanios, E. A., Stretcu, O., Neubig, G., Póczos, B., and Mitchell, T. M. Competence-based curriculum learning for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 1162–1172.

Qin, Y., Wang, X., Zhang, Z., and Zhu, W. Graph differentiable architecture search with structure learning. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 16860–16872, 2021.

Raghu, A., Raghu, M., Kornblith, S., Duvenaud, D., and Hinton, G. E. Teaching with commentaries. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Saxena, S., Tuzel, O., and DeCoste, D. Data parameters: A new family of parameters for learning a differentiable curriculum. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 11093–11103, 2019.

Shelhamer, E., Mahmoudieh, P., Argus, M., and Darrell, T. Loss is its own reward: Self-supervision for reinforcement learning. In *ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*, 2017.

Shi, B., Hoffman, J., Saenko, K., Darrell, T., and Xu, H. Auxiliary task reweighting for minimum-data learning. In *NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012.*, pp. 2960–2968, 2012.

Song, W., Shi, C., Xiao, Z., Duan, Z., Xu, Y., Zhang, M., and Tang, J. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, pp. 1161–1170. ACM, 2019.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*, pp. 751–759. The Association for Computational Linguistics, 2010.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011.

Wang, M. and Deng, W. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

Wang, X., Pham, H., Michel, P., Anastasopoulos, A., Carbonell, J. G., and Neubig, G. Optimizing data usage via differentiable rewards. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9983–9995. PMLR, 2020.

Wang, X., Chen, H., and Zhu, W. Multimodal disentangled representation for recommendation. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6. IEEE, 2021a.

Wang, X., Chen, Y., and Zhu, W. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021b.

Wei, Y., Liang, X., Chen, Y., Shen, X., Cheng, M., Feng, J., Zhao, Y., and Yan, S. STC: A simple to complex framework for weakly-supervised semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(11):2314–2320, 2017.

Wen, H., Zhang, J., Wang, Y., Lv, F., Bao, W., Lin, Q., and Yang, K. Entire space multi-task modeling via post-click behavior decomposition for conversion rate prediction. In *SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pp. 2377–2386, 2020.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Zhang, Z., Zhang, Z., Wang, X., and Zhu, W. Learning to solve travelling salesman problem with hardness-adaptive curriculum. *CoRR*, abs/2204.03236, 2022.

Zhao, H., Stretcu, O., Smola, A. J., and Gordon, G. J. Efficient multitask feature and relationship learning. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*, volume 115 of *Proceedings of Machine Learning Research*, pp. 777–787. AUAI Press, 2019.

Zhu, F., Zhu, Y., Chang, X., and Liang, X. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 10009–10019, 2020.

Zhu, W., Cui, P., Wang, Z., and Hua, G. Multimedia big data computing. *IEEE multimedia*, 22(3):96–c3, 2015.

Zhu, W., Wang, X., and Xie, P. Self-directed machine learning. *CoRR*, abs/2201.01289, 2022.

A. Appendix

A.1. Derivation of the Implicit Gradient

We present the derivation of the implicit gradient here. According to the Cauchy Implicit Function Theorem, if there exists one point (θ_0, β_0) where $\nabla_\theta L_t(\theta, \beta) = 0$ and the regularity conditions are satisfied, then within the neighborhood of (θ_0, β_0), there exists a implicit function $\theta^*(\beta)$ s.t.$\nabla_\theta L_t(\theta, \beta) = 0|_{\beta, \theta^*(\beta)}$. Assuming that $\nabla^2_\theta L_t(\theta^*, \beta)$ is positive definite, we have the following derivation,

$$\nabla_\theta L_t(\theta^*(\beta), \beta) = 0, \tag{10}$$

$$\nabla^2_\theta L_t(\theta^*, \beta)\nabla_\beta \theta^* + \nabla_\beta \nabla_{\theta^*} L_t(\theta^*, \beta) = 0, \tag{11}$$

$$\nabla_\beta \theta^* = -(\nabla^2_\theta L_t(\theta^*, \beta))^{-1} \nabla_\beta \nabla_\theta L_t(\theta^*, \beta) \tag{12}$$

From Eq.(10) to Eq.(11), we take the derivative to β on both sides of the Eq.(10). By assuming that $\nabla^2_\theta L_t(\theta^*, \beta)$ is positive definite, $\nabla^2_\theta L_t(\theta^*, \beta)$ will have an inverse so we can obtain the implicit gradient in Eq.(12).

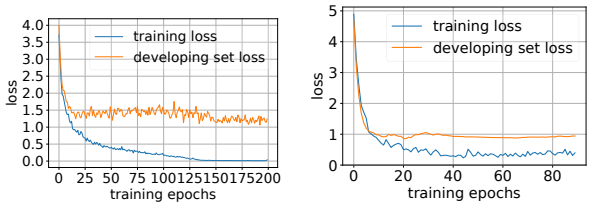
A.2. Computational Complexity Analysis

We provide the computational time complexity of different methods as follows. Denoting lower step as M, auxiliary task number as N, the truncated Neumann series number as K, we regard the STL(single task learning) as baseline(assuming its time complexity as 1). The complexity of different methods to obtain their optimal weights are as follows. (1)STL:1; (2)NAL: 2^N(assuming 2 candidates for each task weight).; (3)Uncert:1; (4):N+1, considering that most of the compuational time comes from the backward process; (5)AuxL&ours:1+(K+3)/M, typical value in our paper is 1.3, after each M step lower optimization, one upper optimization needs (3+K) backward. Therefore, the time of our method is (M+K+3)/M that of STL baseline.

A.3. More Learning behavior

Loss behavior during bi-level optimization Our proposed method optimizes the task learning model and the joint scheduler in a bi-level iterative way. It is important that the loss on the training set and the developing set could stably descend. We visualize the loss of the target task on the training set and the developing set in the CUB and CIFAR100 dataset and present them in Figure 5. The target training loss and the loss on the developing set are optimized in a stable way.

The learned adaptive filters To distinguish whether a sample is noisy or not, we design a filter based on the sample loss for each task. We present the final learned functions for the target task on the CUB dataset in both the supervised and 0.2 corrupted ratio settings in Figure 6. We can observe that in the setting of 0.2 ratio of corrupted labels, larger normalized loss will result in lower schedule weights. This



(a) CIFAR100 (b) CUB

Figure 5. Loss behavior on the CUB and CIFAR100 experiment

learned function also fits the results in (Arazo et al., 2019) where noisy samples usually result in larger value of loss. However, in the supervised setting, the weights all samples are close to 1, indicating that the labels for the samples are almost correct. The learned different filter functions for the main task suggests that under different settings, the most suitable filters are different. It is necessary to give the filters learnable parameters $\{a_k\}$ and $\{b_k\}$ make them automatically adaptive to different scenarios.

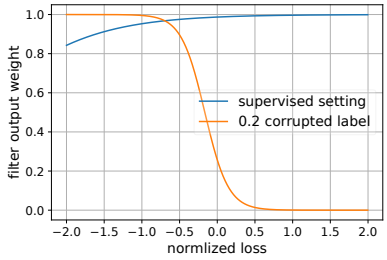


Figure 6. Target task filter function for supervised and corrupted settings.

A.4. Experimental Details

Model Structure In our experiments, we adopt the most widely used hard parameter sharing model as the task learning model. In the CIFAR100, CUB and Pet datasets, we use the ResNet18 (He et al., 2016) as the backbone which is used as the aforementioned $f_{k,enc}$. Particularly, to fit the smaller image size in CIFAR100, we replace the original 7x7 convolution kernel to 3x3 and remove the maxpooling layer in the first block. For the CIFAR10 dataset, we adopt a 4-layer ConvNet as the backbone. Each layer is composed of the [Conv2d, BatchNormalization, ReLU] components. After the 2nd and the 4th layer, there is a 2×2 Maxpooling layer. The number of output channel for all the layers is set to 32, and the kernel size for the Conv2d is 3×3 and the stride is 1. For the recommendation task, AutoINT (Song et al., 2019) is the adopted backbone composed of 4 transformer encoder block with head number 4 and embedding dimension 16. As for the task heads, we adopt the Multilayer Perceptron(MLP) with number of layers searched in

$\{1,2\}$ by the STL baseline.

Hyper-parameter About the optimization process, in the CIFAR100 and CIFAR10 dataset, we optimize θ using SGD with initial learning rate searched from $\{0.01,0.02,0.05\}$ and a cosine annealing scheduler, and in the CUB, Pet, MovieLens-1M dataset we use Adam (Kingma & Ba, 2014) with learning rate searched from $\{1e\text{-}3,1e\text{-}4\}$. To optimize β, we use SGD with learning rate from $\{1e\text{-}2,1e\text{-}3\}$. Additionally, M in the Algorithm is searched from $\{10, 20, 50\}$ and K is searched from $\{3,5\}$ considering the performance and computational cost as recommended in (Lorraine et al., 2020). The datasets are split into training, validation and test set as officially recommended. We sample 200 samples from the validation set to form D_{dev} with stratified sampling. For the baselines that do not use D_{dev} as guidance, we add these samples to their training set for more fair comparison. The running epoch is 200 for CIFAR100 and Pet, 100 for CUB, 15 for MovieLens-1M. Additionally, the weights for baseline NAL is searched from $\{0.1,1.0,10.0\}$.

A.5. The Implementation of N-JTDS algorithm

We present the algorithm implementation of the N-JTDS baseline in Algorithm 2. This algorithm is a naive implementation of the joint task and sample scheduling for auxiliary learning. The scheduler does not rely on any information to predict the schedule but assigns one learnable parameter as the weight to each data sample under each task. The gradient of these weights are also calculated in the same implicit differentiation way as our proposed JTDS. We also denote all the weights parameters $\beta = \{w_{ik}\}_{i=1,k\in U}^{m}$.

Algorithm 2 N-JTDS for Auxiliary Learning

Input: D_t, D_{dev}, D_v, $interval$, η_1, η_2, K, M;
Initialize θ, β, θ_{opt}, $t = 0$, $error_v = 1.0$;
while not converged **do**
  // lower optimization
  for $j = 1$ **to** M **do**
    for $(x_i^t, y_{ik}^t) \in D_t$ **do**
      $l_{ik} = l_k(f_k(x_i^t), y_{ik}^t; \theta)$;
    end for
    $\theta \leftarrow \theta - \eta_1 \sum_{k\in U} \sum_{i=1}^{m} w_{ik} \nabla_\theta l_{ik}$;
  end for
  $t \leftarrow t + 1$;
  // record the best parameters on D_v
  if $t\%interval == 0$ **then**
    $error = E_v(f_G, D_v; \theta)$;
    if $error < error_v$ **then**
      $error_v = error, \theta_{opt} = \theta$;
    end if
  end if
  // upper optimization
  $L_{dev} = \sum_{i=1}^{r} l_G(f_G(x_i^{dev}), y_{iG}^{dev}; \theta)$;
  for $(x_i^t, y_{ik}^t) \in D_t$ **do**
    $l_{ik} = l_k(f_k(x_i^t), y_{ik}^t; \theta)$,
  end for
  $L_t = \sum_{k\in U} \sum_{i=1}^{m} w_{ik} * l_{ik}$;
  $\beta \leftarrow \beta + \eta_2 \nabla_\theta L_{dev} \cdot \sum_{i=0}^{K} (I - \nabla_\theta^2 L_t)^i \cdot \nabla_\beta \nabla_\theta L_t$;
end while
Return θ_{opt}