Auxiliary Task Reweighting for Minimum-data Learning

Baifeng Shi
Peking University
bfshi@pku.edu.cn

Judy Hoffman
Georgia Institute of Technology
judy@gatech.edu

Kate Saenko
Boston University & MIT-IBM Watson AI Lab
saenko@bu.edu

Trevor Darrell, Huijuan Xu
University of California, Berkeley
{trevor, huijuan}@eecs.berkeley.edu

Abstract

Supervised learning requires a large amount of training data, limiting its application where labeled data is scarce. To compensate for data scarcity, one possible method is to utilize auxiliary tasks to provide additional supervision for the main task. Assigning and optimizing the importance weights for different auxiliary tasks remains an crucial, and largely understudied, research question. In this work, we propose a method to automatically reweight auxiliary tasks in order to reduce the data requirement on the main task. Specifically, we formulate the weighted likelihood function of auxiliary tasks as a surrogate prior for the main task. By then adjusting the auxiliary task weights to minimize the divergence between the surrogate prior and the true prior of the main task, we obtain a more accurate prior estimation. This minimizes the required amount of training data for the main task and avoids a costly grid search. In multiple experimental settings (*e.g.* semi-supervised learning, multi-label classification), we demonstrate that our algorithm can effectively utilize limited labeled data with the benefit of auxiliary tasks compared with previous task reweighting methods. We also show that under extreme cases with only a few extra examples (*e.g.* few-shot domain adaptation), our algorithm results in significant improvement over the baseline.

1 Introduction

Supervised deep learning methods typically require an enormous amount of labeled data, which for many applications, is difficult, time-consuming, expensive, or even impossible, to collect. As a result, there is a significant research effort devoted to efficient learning with limited labeled data, including semi-supervised learning [48, 56], transfer learning [57], few-shot learning [10], domain adaptation [58], and representation learning [49].

Among these different approaches, auxiliary tasks are widely used to address the lack of data by providing additional supervision, *i.e.* using the same data or auxiliary data for a different learning task during the training procedure. Auxiliary tasks are either collected from related tasks, domains where there is abundant data [58], or manually designed to fit the latent data structure [55, 64]. Training with auxiliary tasks has been shown to achieve better generalization [2], and is therefore widely

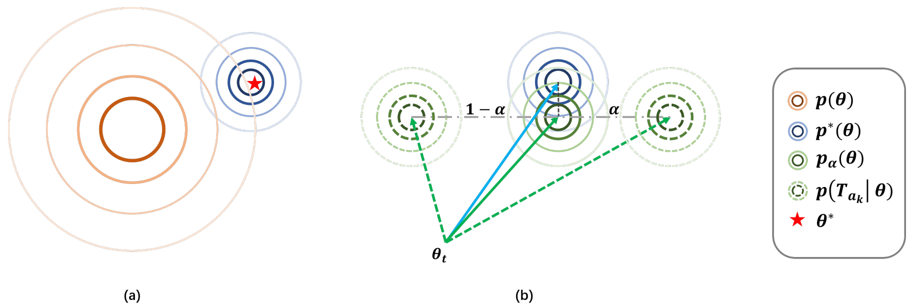


Figure 1: Learning with minimal data through auxiliary task reweighting. (a) An ordinary prior $p(\theta)$ over model parameters contains little information about the true prior $p^*(\theta)$ and the optimal parameter θ^*. (b) Through a weighted combination of distributions induced by data likelihood $p(\mathcal{T}_{a_k}|\theta)$ of different auxiliary tasks, we find the optimal surrogate prior $p_\alpha(\theta)$ which is closest to the true prior.

used in many applications, *e.g.* semi-supervised learning [64], self-supervised learning [49], transfer learning [57], and reinforcement learning [30].

Usually both the main task and auxiliary task are jointly trained, but only the main task's performance is important for the downstream goals. The auxiliary tasks should reduce the amount of labeled data needed to achieve a given performance for the main task. However, this has proven to be a difficult selection problem as certain, seemingly related tasks yield little or no improvement. One simple task selection strategy is to compare the main task performance when training with each auxiliary task separately [63]. However, this requires an exhaustive enumeration of all candidate tasks, which is prohibitively expensive when the candidate pool is large. Furthermore, the individual tasks may behave unexpectedly when combined together for final training. Another strategy, is to train all auxiliary tasks together in a single pass and then use an evaluation technique or algorithm to automatically determine the importance weight for each task. There are several works in this direction [5, 11, 14, 37], but they either only filter out unrelated tasks without further differentiating among related ones, or have a focused motivation (*e.g.* faster training) limiting their general use.

In this work, we propose a method to adaptively reweight auxiliary tasks on the fly during joint training so that the data requirement on the main task is minimized. We start from a key insight: *we can reduce the data requirement by choosing a high-quality prior*. Then we formulate the parameter distribution induced by the auxiliary tasks' likelihood as a surrogate prior for the main task. By adjusting the auxiliary task weights, the divergence between the surrogate prior and the true prior of the main task is minimized. In this way, the data requirement on the main task is reduced under high quality surrogate prior. Specifically, due to the fact that minimizing the divergence is intractable, we turn the optimization problem into minimizing the distance between gradients of the main loss and the auxiliary losses, which allows us to design a practical, light-weight algorithm. We show in various experimental settings that our method can make better use of labeled data and effectively reduce the data requirement for the main task. Surprisingly, we find that very little labeled data (*e.g.* 1 image per class) is enough for our algorithm to bring a substantial improvement over unsupervised and few-shot baselines.

2 Learning with Minimal Data through Auxiliary Task Reweighting

Suppose we have a main task with training data $\mathcal{T}_m$ (including labels), and K different auxiliary tasks with training data $\mathcal{T}_{a_k}$ for k-th task, where $k = 1, \cdots, K$. Our model contains a shared backbone with parameter θ, and different heads for each task. Our goal is to find the optimal parameter θ^* for the main task, using data from main task as well as auxiliary tasks. Note that we care about performance on the main task and auxiliary tasks are only used to help train a better model on main task (*e.g.* when we do not have enough data on the main task). In this section, we discuss how to learn with minimal data on main task by learning and reweighting auxiliary tasks.

2.1 How Much Data Do We Need: Single-Task Scenario

Before discussing learning from multiple auxiliary tasks, we first start with the single-task scenario. When there is only one single task, we normally train a model by minimizing the following loss:

$$\mathcal{L}(\theta) = -\log p(\mathcal{T}_m|\theta) - \log p(\theta) = -\log(p(\mathcal{T}_m|\theta) \cdot p(\theta)), \tag{1}$$

where $p(\mathcal{T}_m|\theta)$ is the likelihood of training data and $p(\theta)$ is the prior. Usually, a relatively weak prior (*e.g.* Gaussian prior when using weight decay) is chosen, reflecting our weak knowledge about the true parameter distribution, which we call 'ordinary prior'. Meanwhile, we also assume there exists an unknown 'true prior' $p^*(\theta)$ which the optimal parameter θ^* is actually sampled from. This true prior is normally more selective and informative (*e.g.* having a small support set) (See Fig. 1(a)) [6].

Now our question is, how much data do we need to learn the task. Actually the answer depends on the choice of the prior $p(\theta)$. If we know the informative 'true prior' $p^*(\theta)$, only a few data items are needed to localize the best parameters θ^* within the prior. However, if the prior is rather weak, we have to search θ in a larger space, which needs more data. Intuitively, the required amount of data is related to the divergence between $p(\theta)$ and $p^*(\theta)$: the closer they are, the less data we need.

In fact, it has been proven [6] that the expected amount of information needed to solve a single task is

$$\mathcal{I} = D_{\mathrm{KL}}(p^* \parallel p) + H(p^*), \tag{2}$$

where $D_{\mathrm{KL}}(\cdot \parallel \cdot)$ is Kullback–Liebler divergence, and $H(\cdot)$ is the entropy. This means we can reduce the data requirement by choosing a prior closer to the true prior p^*. Suppose $p(\theta)$ is parameterized by α, *i.e.*, $p(\theta) = p_\alpha(\theta)$, then we can minimize data requirement by choosing α that satisfies:

$$\min_\alpha D_{\mathrm{KL}}(p^* \parallel p_\alpha). \tag{3}$$

However, due to our limited knowledge about the true prior p^*, it is unlikely to manually design a family of p_α that has a small value in (3). Instead, we will show that we can define p_α implicitly through auxiliary tasks, utilizing their natural connections to the main task.

2.2 Auxiliary-Task Reweighting

When using auxiliary tasks, we optimize the following joint-training loss:

$$\mathcal{L}(\theta) = -\log p(\mathcal{T}_m|\theta) - \sum_{k=1}^{K} \alpha_k \log p(\mathcal{T}_{a_k}|\theta) = -\log(p(\mathcal{T}_M|\theta) \cdot \prod_{k=1}^{K} p^{\alpha_k}(\mathcal{T}_{a_k}|\theta)), \tag{4}$$

where auxiliary losses are weighted by a set of task weights $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_K)$, and added together with the main loss. By comparing (4) with single-task loss (1), we can see that we are implicitly using $p_{\boldsymbol{\alpha}}(\theta) = \frac{1}{Z(\boldsymbol{\alpha})} \prod_{k=1}^{K} p^{\alpha_k}(\mathcal{T}_{a_k}|\theta)$ as a 'surrogate' prior for the main task, where $Z(\boldsymbol{\alpha})$ is the normalization term (partition function). Therefore, as discussed in Sec. 2.1, if we adjust task weights $\boldsymbol{\alpha}$ towards

$$\min_{\boldsymbol{\alpha}} D_{\mathrm{KL}}(p^*(\theta) \parallel \frac{1}{Z(\boldsymbol{\alpha})} \prod_{k=1}^{K} p^{\alpha_k}(\mathcal{T}_{a_k}|\theta)), \tag{5}$$

then the data requirement on the main task can be minimized. This implies an automatic strategy of task reweighting. Higher weights can be assigned to the auxiliary tasks with more relevant information to the main task, namely the parameter distribution of the tasks is closer to that of the main task. After taking the weighted combination of auxiliary tasks, the prior information is maximized, and the main task can be learned with minimal additional information (data). See Fig. 1(b) for an illustration.

2.3 Our Approach

In Sec. 2.2 we have discussed about how to minimize the data requirement on the main task by reweighting and learning auxiliary tasks. However, the objective in (5) is hard to optimize directly due to a few problems:

- **True Prior (P1)**: We do not know the true prior p^* in advance.

- **Samples (P2)**: KL divergence is in form of an expectation, which needs samples to estimate. However, sampling from a complex distribution is non-trivial.
- **Partition Function (P3)**: Partition function $Z(\boldsymbol{\alpha}) = \int \prod_{k=1}^{K} p^{\alpha_k}(\mathcal{T}_{a_k}|\theta)d\theta$ is given by an intractable integral, preventing us from getting the accurate density function $p_{\boldsymbol{\alpha}}$.

To this end, we use different tools or approximations to design a practical algorithm, meanwhile keeping its validity and effectiveness from both theoretical and empirical aspects, as presented below.

True Prior (P1) In the original optimization problem (5), we are minimizing

$$D_{\mathrm{KL}}(p^*(\theta) \parallel p_{\boldsymbol{\alpha}}(\theta)) = E_{\theta\sim p^*} \log \frac{p^*(\theta)}{p_{\boldsymbol{\alpha}}(\theta)}, \tag{6}$$

which is the expectation of $\log \frac{p^*(\theta)}{p_{\boldsymbol{\alpha}}(\theta)}$ w.r.t. $p^*(\theta)$. The problem is, $p^*(\theta)$ is not accessible. However, we can notice that for each θ^* sampled from prior p^*, it is likely to give a high data likelihood $p(\mathcal{T}_m|\theta^*)$, which means $p^*(\theta)$ is 'covered' by $p(\mathcal{T}_m|\theta)$, *i.e.*, $p(\mathcal{T}_m|\theta)$ has high density both in the support set of $p^*(\theta)$, and in some regions outside. Thus we propose to minimize $D_{\mathrm{KL}}(p^m(\theta) \parallel p_{\boldsymbol{\alpha}}(\theta))$ instead of $D_{\mathrm{KL}}(p^*(\theta) \parallel p_{\boldsymbol{\alpha}}(\theta))$, where $p^m(\theta)$ is the parameter distribution induced by data likelihood $p(\mathcal{T}_m|\theta)$, *i.e.*, $p^m(\theta) \propto p(\mathcal{T}_m|\theta)$. Furthermore, we propose to take the expectation w.r.t. $\frac{1}{Z'(\boldsymbol{\alpha})}p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)$ instead of $p^m(\theta)$ due to convenience of sampling while optimizing the joint loss (see **P2** for more details). Then our objective becomes

$$\min_{\boldsymbol{\alpha}} E_{\theta\sim p^J} \log \frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)}, \tag{7}$$

where $p^J(\theta) = \frac{1}{Z'(\boldsymbol{\alpha})}p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)$, and $Z'(\boldsymbol{\alpha})$ is the normalization term.

Now we can minimize (7) as a feasible surrogate for (5). However, minimizing (7) may end up with a suboptimal $\boldsymbol{\alpha}$ for (5). Due to the fact that $p^m(\theta)$ also covers some 'overfitting area' other than $p^*(\theta)$, we may push $p_{\boldsymbol{\alpha}}(\theta)$ closer to the overfitting area instead of $p^*(\theta)$ by minimizing (7). But we prove that, under some mild conditions, if we choose $\boldsymbol{\alpha}$ that minimizes (7), the value of (5) is also bounded near the optimal value:

Theorem 1. *(Informal and simplified version) Let us denote the optimal weights for* (5) *and* (7) *by* $\boldsymbol{\alpha}^*$ *and* $\hat{\boldsymbol{\alpha}}$, *respectively. Assume the true prior* $p^*(\theta)$ *has a small support set* S. *Let* $\gamma = \max_{\boldsymbol{\alpha}} \int_{\theta\notin S} p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)d\theta$ *be the maximum of integral of* $p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)$ *out of* S, *then we have*

$$D_{\mathrm{KL}}(p^* \parallel p_{\hat{\boldsymbol{\alpha}}}) \leq D_{\mathrm{KL}}(p^* \parallel p_{\boldsymbol{\alpha}^*}) \;+\; C\gamma^2 \;-\; C'\gamma^2 \log \gamma. \tag{8}$$

The formal version and proof can be found in Appendix. Theorem 1 states that optimizing (7) can also give a near-optimal solution for (5), as long as γ is small. This condition holds when $p^m(\theta)$ and $p_{\boldsymbol{\alpha}}(\theta)$ do not reach a high density at the same time outside S. This is reasonable because overfitted parameter of main task (*i.e.*, θ giving a high training data likelihood outside S) is highly random, depending on how we sample the training set, thus is unlikely to meet the optimal parameters of auxiliary tasks. In practice, we also find this approximation gives a robust result (Sec. 3.3).

Samples (P2) To estimate the objective in (7), we need samples from $p^J(\theta) = \frac{1}{Z'(\boldsymbol{\alpha})}p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)$. Apparently we cannot sample from this complex distribution directly. However, we notice that p^J is what we optimize in the joint-training loss (4), *i.e.*, $\mathcal{L}(\theta) \propto -\log p^J(\theta)$. To this end, we use the tool of Langevin dynamics [46, 60] to sample from p^J while optimizing the joint-loss (4). Specifically, at the t-th step of SGD, we inject a Gaussian noise with a certain variance into the gradient step:

$$\Delta\theta_t = -\epsilon_t \nabla\mathcal{L}(\theta) + \eta_t, \tag{9}$$

where ϵ_t is the learning rate, and $\eta_t \sim N(0, 2\epsilon_t)$ is a Guassian noise. With the injected noise, θ_t will converge to samples from p^J, which can then be used to estimate (7). In practice, we inject noise in early epochs to sample from p^J and optimize $\boldsymbol{\alpha}$, and then return to regular SGD once $\boldsymbol{\alpha}$ has converged. Note that we do not anneal the learning rate as in [60] because we find in practice that stochastic gradient noise is negligible compared with injected noise (see Appendix).

Algorithm 1 ARML (**A**uxiliary Task **R**eweighting for **M**inimum-data **L**earning)

```
Input: main task data $\mathcal{T}_m$, auxiliary task data $\mathcal{T}_{a_k}$, initial parameter $\theta_0$, initial task weights $\boldsymbol{\alpha}$
Parameters: learning rate of $t$-th iteration $\epsilon_t$, learning rate for task weights $\beta$

for iteration $t = 1$ to $T$ do
  if $\boldsymbol{\alpha}$ has not converged then
    $\theta_t \leftarrow \theta_{t-1} - \epsilon_t(-\nabla \log p(\mathcal{T}_m|\theta_{t-1}) - \sum_{k=1}^K \alpha_k \nabla \log p(\mathcal{T}_{a_k}|\theta_{t-1})) + \eta_t$
    $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \beta \nabla_{\boldsymbol{\alpha}} \|\nabla \log p(\mathcal{T}_m|\theta_t) - \sum_{k=1}^K \alpha_k \nabla \log p(\mathcal{T}_{a_k}|\theta_t)\|_2^2$
    Project $\boldsymbol{\alpha}$ back into $\mathcal{A}$
  else
    $\theta_t \leftarrow \theta_{t-1} - \epsilon_t(-\nabla \log p(\mathcal{T}_m|\theta_{t-1}) - \sum_{k=1}^K \alpha_k \nabla \log p(\mathcal{T}_{a_k}|\theta_{t-1}))$
  end if
end for
```

Partition Function (P3) To estimate (7), we need the exact value of surrogate prior $p_{\boldsymbol{\alpha}}(\theta) = \frac{1}{Z(\boldsymbol{\alpha})} \prod_{k=1}^K p^{\alpha_k}(\mathcal{T}_{a_k}|\theta)$. Although we can easily calculate the data likelihood $p(\mathcal{T}_{a_k}|\theta)$, the partition function $Z(\boldsymbol{\alpha})$ is intractable. The same problem also occurs in model estimation [21], Bayesian inference [45], *etc*. A common solution is to use score function $\nabla \log p_{\boldsymbol{\alpha}}(\theta)$ as a substitution of $p_{\boldsymbol{\alpha}}(\theta)$ to estimate relationship with other distributions [26, 28, 39]. For one reason, score function can uniquely decide the distribution. It also has other nice properties. For example, the divergence defined on score functions (also know as *Fisher divergence*)

$$F(p \parallel q) = E_{\theta \sim p} \|\nabla \log p(\theta) - \nabla \log q(\theta)\|_2^2 \tag{10}$$

is stronger than many other divergences including KL divergence, Hellinger distance, *etc*. [26, 38]. Most importantly, using score function can obviate estimation of partition function which is constant w.r.t. θ. To this end, we propose to minimize the distance between score functions instead, and our objective finally becomes

$$\min_{\boldsymbol{\alpha}} E_{\theta \sim p^J} \|\nabla \log p(\mathcal{T}_m|\theta) - \nabla \log p_{\boldsymbol{\alpha}}(\theta)\|_2^2. \tag{11}$$

Note that $\nabla \log p^m(\theta) = \nabla \log p(\mathcal{T}_m|\theta)$. In Appendix we show that under mild conditions the optimal solution for (11) is also the optimal or near-optimal $\boldsymbol{\alpha}$ for (5) and (7) . We find in practice that optimizing (11) generally gives optimal weights for minimum-data learning.

2.4 Algorithm

Now we present the final algorithm of auxiliary task reweighting for minimum-data learning (ARML). The full algorithm is shown in Alg. 1. First, our objective is (11). Until $\boldsymbol{\alpha}$ converges, we use Langevin dynamics (9) to collect samples at each iteration, and then use them to estimate (11) and update α. Additionally, we only search $\boldsymbol{\alpha}$ in an affine simplex $\mathcal{A} = \{\boldsymbol{\alpha} | \sum_k \alpha_k = K;\ \alpha_k \geq 0, \forall k\}$ to decouple task reweighting from the global weight of auxiliary tasks [11]. Please also see Appendix A.4 for details on the algorithm implementation in practice.

3 Experiments

For experiments, we test effectiveness and robustness of ARML under various settings. This section is organized as follows. First in Sec. 3.1, we test whether ARML can reduce data requirement in different settings (semi-supervised learning, multi-label classification), and compare it with other reweighting methods. In Sec. 3.2, we study an extreme case: based on an unsupervised setting (*e.g.* domain generalization), if a little extra labeled data is provided (*e.g.* 1 or 5 labels per class), can ARML maximize its benefit and bring a non-trivial improvement over unsupervised baseline and other few-shot algorithms? Finally in Sec. 3.3, we test ARML's robustness under different levels of data scarcity and validate the rationality of approximation we made in Sec. 2.3.

3.1 ARML can Minimize Data Requirement

To get started, we show that ARML can minimize data requirement under two realist settings: semi-supervised learning and multi-label classification. we consider the following task reweighting methods for comparison: (i) **Uniform (baseline)**: all weights are set to 1, (ii) **AdaLoss** [25]: tasks are reweighted based on uncertainty, (iii) **GradNorm** [11]: balance each task's gradient norm, (iv) **CosineSim** [14]: tasks are filtered out when having negative cosine similarity $\cos(\nabla \log p(\mathcal{T}_{a_k}|\theta), \nabla \log p(\mathcal{T}_m|\theta))$, (v) **OL_AUX** [37]: tasks have higher weights when the gradient inner product $\nabla \log p(\mathcal{T}_{a_k}|\theta)^T \nabla \log p(\mathcal{T}_m|\theta)$ is large. Besides, we also compare with grid search as an 'upper bound' of ARML. Since grid search is extremely expensive, we only compare with it when the task number is small (*e.g.* $K = 2$).

Semi-supervised Learning (SSL) In SSL, one generally trains classifier with certain percentage of labeled data as the main task, and at the same time designs different losses on unlabeled data as auxiliary tasks. Specifically, we use *Self-supervised Semi-supervised Learning* (S4L) [64] as our baseline algorithm. S4L uses self-supervised methods on unlabeled part of training data, and trains classifier on labeled data as normal. Following [64], we use two kinds of self-supervised methods: *Rotation* and *Exemplar-MT*. In *Rotation*, we rotate each image by $[0^\circ, 90^\circ, 180^\circ, 270^\circ]$ and ask the network to predict the angle. In *Exemplar-MT*, the model is trained to extract feature invariant to a wide range of image transformations. Here we use random flipping, gaussian noise [9] and Cutout [13] as data augmentation. During training, each image is randomly augmented, and then features of original image and augmented image are encouraged to be close.

Table 1: Test error of semi-supervised learning on CIFAR-10 and SVHN. From top to bottom: purely-supervised method, state-of-the-art semi-supervised methods, and S4L with different reweighting schemes. * means multiple runs are needed.

	CIFAR-10 (4000 labels)	SVHN (1000 labels)
Supervised	20.26 ± .38	12.83 ± .47
Π-Model [34]	16.37 ± .63	7.19 ± .27
Mean Teacher [56]	15.87 ± .28	5.65 ± .47
VAT [48]	13.86 ± .27	5.63 ± .20
VAT + EntMin [19]	**13.13** ± .39	**5.35** ± .19
Pseudo-Label [35]	17.78 ± .57	7.62 ± .29
S4L (Uniform)	15.67 ± .29	7.83 ± .33
S4L + AdaLoss	21.06 ± .17	11.53 ± .39
S4L + GradNorm	14.07 ± .44	7.68 ± .13
S4L + CosineSim	15.03 ± .31	7.02 ± .25
S4L + OL_AUX	16.07 ± .51	7.82 ± .32
S4L + GridSearch*	13.76 ± .22	6.07 ± .17
S4L + ARML (ours)	**13.68** ± .35	**5.89** ± .22

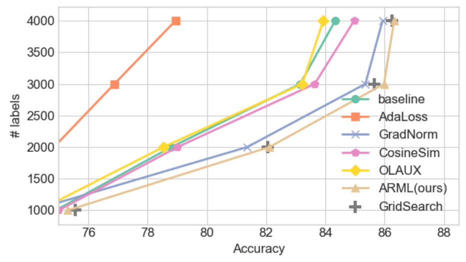


Figure 2: Amount of labeled data required to reach certain accuracy on CIFAR-10.

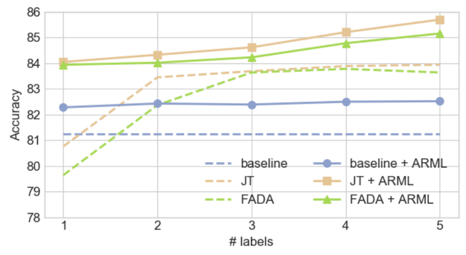


Figure 3: Accuracy of multi-source domain generalization with Art as target.

Based on S4L, we use task reweighting to adjust the weights for different self-supervised losses. Following the literature [48, 56], we test on two widely-used benchmarks: CIFAR-10 [33] with 4000 out of 45000 images labeled, and SVHN [47] with 1000 out of 65932 images labeled. We report test error of S4L with different reweighting schemes in Table 1, along with other SSL methods. We notice that, on both datasets, with the same amount of labeled data, ARML makes a better use of the data than uniform baseline as well as other reweighting methods. Remarkably, with only one pass, ARML is able to find the optimal weights while GridSearch needs multiple runs. S4L with our ARML applied is comparable to other state-of-the-art SSL methods. Notably, we only try *Rotation* and *Exemplar-MT*, while exploring more auxiliary tasks could further benefit the main task, we leave it for future study.

To see whether ARML can consistently reduce data requirement, we also test the amount of data required to reach different accuracy on CIFAR-10. As shown in Fig. 2, with ARML applied, we only need about half of labels to reach a decent performance. This also agrees with the results of GridSearch, showing the maximum improvement from auxiliary tasks during joint training.

Table 2: Test error of main task on CelebA.

	Test Error
Baseline	$6.70 \pm .18$
AdaLoss [25]	$7.21 \pm .11$
GradNorm [11]	$6.44 \pm .07$
CosineSim [14]	$6.51 \pm .14$
OL_AUX [37]	$6.32 \pm .17$
ARML (ours)	**5.97** $\pm .18$

Table 3: Top 5 related / unrelated attributes (auxiliary tasks) to the target attribute (main task) on CelebA.

main task	most related tasks	least related tasks
5_o_Clock_Shadow	Mustache	Mouth_Slightly_Open
	Bald	Male
	Sideburns	Attractive
	Rosy_Cheeks	Heavy_Makeup
	Goatee	Smiling

Multi-label Classification (MLC) We also test our method in MLC. We use the CelebA dataset [40]. It contains 200K face images, each labeled with 40 binary attributes. We cast this into a MLC problem, where we randomly choose one target attribute as the main classification task, and other 39 as auxiliary tasks. To simulate our data-scarce setting, we only use 1% labels for main task.

We test different reweighting methods and list the results in Table 2. With the same amount of labeled data, ARML can help find better and more generalizable model parameters than baseline as well as other reweighting methods. This also implies that ARML has a consistent advantage even when handling a large number of tasks. For a further verification, we also check if the learned relationship between different face attributes is aligned with human's intuition. In Table 3, we list the top 5 auxiliary tasks with the highest weights, and also the top 5 with the lowest weights. As we can see, ARML has automatically picked attributes describing facial hair (Mustache, Sideburns, Goatee), which coincides with the main task 5_o_Clock_Shadow, another kind of facial hair. On the other hand, the tasks with low weights seem to be unrelated to the main task. This means ARML can actually learn the task relationship that matches our intuition.

3.2 ARML can Benefit Unsupervised Learning at Minimal Cost

In Sec. 3.1, we use ARML to reweight tasks and find a better prior for main task in order to compensate for data scarcity. Then one may naturally wonder if this still works under situations where the main task has no labeled data at all (*e.g.* unsupervised learning). In fact, this is a meaningful and significant question, not only because unsupervised learning is one of the most important problems in the community, but also because using auxiliary tasks is a mainstream of unsupervised learning methods [8, 18, 49]. Intuitively, as long as the family of prior $p_{\boldsymbol{\alpha}}(\theta)$ is strong enough (which is determined by auxiliary tasks), we can always find a prior that gives a good model even without label information. However, if we want to use ARML to find the prior, at least *some* labeled data is needed to estimate the gradient for main task (Eq. (11)). Then the question becomes, how minimum data does ARML need to find a proper set of weights? More specifically, can we use as little data as possible (*e.g.* 1 or 5 labeled images per class) to make substantial improvements?

To answer the question, we conduct experiments in domain generalization, a well-studied unsupervised problem. In domain generalization, there is a target domain with no data (labeled or unlabeled), and multiple source domains with plenty data. One aims to train a model on source domains (auxiliary tasks) and transfer it to the target domain (main task). To use ARML, we relax the restriction a little by adding N_m extra labeled images for target domain, where $N_m = 1, \cdots, 5$. This slightly relaxed setting is also known as few-shot domain adaptation (FSDA) which was also studied in [44]. We also add their results into comparison. For dataset, we use a common benchmark of PACS [36] which contains four distinct domains of Photo, Art, Cartoon and Sketch. We pick each one as target domain and other three as source domains which are reweighted by ARML.

We first set $N_m = 5$ to see the results (Table 4). Here we include both state-of-the-art domain generalization methods [4, 8, 15] and FSDA methods [44]. Since they are orthogonal to ARML, we apply ARML on both types of methods to see the relative improvements. Let us first look at domain generalization methods. Here the baseline refers to training a model on source domains (auxiliary tasks) and directly testing on target domain (main task). If we use the extra 5 labels to reweight different source domains with ARML, we can make a non-trivial improvement, especially with Sketch as target (4% absolute improvement). Note that in "Baseline + ARML", we update θ using only classification loss on source data (auxiliary loss), and the extra labeled data in the target domain are just for reweighting the auxiliary tasks, which means the improvement completely comes

Table 4: Results of multi-source domain generalization (w/ extra 5 labeled images per class in target domain). We list results with each of four domains as target domain. From top to down: domain generalization methods, FSDA methods and different methods equipped with ARML. JT is short for joint-training. † means the results we reproduced are higher than originally reported.

Method	Extra label	Sketch	Art	Cartoon	Photo
Baseline†	✗	75.34	81.25	77.35	95.93
D-SAM [15]	✗	77.83	77.33	72.43	95.30
JiGen [8]	✗	71.35	79.42	75.25	96.03
Shape-bias [4]	✗	**78.62**	**83.01**	**79.39**	**96.83**
JT	✓	78.52	83.94	**81.36**	97.01
FADA [44]	✓	79.23	83.64	79.39	97.07
Baseline + ARML	✓	79.35	82.52	77.30	95.99
JT + ARML	✓	**80.47**	**85.70**	81.01	**97.22**
FADA + ARML	✓	79.46	85.16	81.23	97.01

from task reweighting. Additionally, JT and FSDA methods also use extra labeled images by adding them into classification loss. If we further use them for task reweighting, then we can make a further improvement and reach a state-of-the-art performance.

We also test performance of ARML with $N_m = 1, \cdots, 5$. As an example, here we use Art as target domain. As shown by Fig. 3, ARML is able to improve the accuracy over different domain generalization or FSDA methods. Remarkably, when $N_m = 1$, although FSDA methods are under-performed, ARML can still bring an improvement of $\sim 4\%$ accuracy. This means ARML can benefit unsupervised domain generalization only with the minimal cost of 1 labeled image per class.

3.3 ARML is Robust to Data Scarcity

Finally, we examine the robustness of our method. Due to the approximation made in Sec. 2.3, ARML may find a suboptimal solution. For example, in the true prior approximation (**P1**), we use $p(\mathcal{T}_m|\theta)$ to replace $p^*(\theta)$. When size of $\mathcal{T}_m$ is large, these two should be close to each other. However, if we have less data, $p(\mathcal{T}_m|\theta)$ may also have high-value region outside $p^*(\theta)$ (*i.e.* 'overfitting' area), which may make the approximation inaccurate. To test robustness of ARML, we check whether ARML can find similar task weights under different levels of data scarcity.

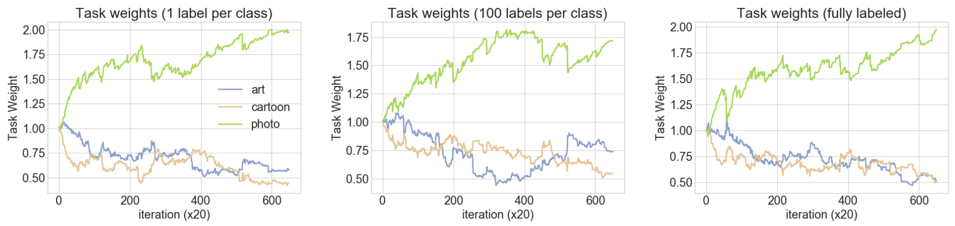


Figure 4: Change of task weights during training under different levels of data scarcity. From left to right: one-shot, partially labeled and fully labeled.

We conduct experiments on multi-source domain generalization with Art as target domain. We test three levels of data scarcity: few-shot (1 label per class), partly labeled (100 labels per class) and fully labeled ($\sim$ 300 labels per class). We plot the change of task weights during training time in Fig. 4. We can see that task weights found by ARML are barely affected by data scarcity, even in few-shot scenario. This means ARML is able to find the optimal weights even with minimal guidance, verifying the rationality of approximation in Sec. 2.3 and the robustness of our method.

4 Related Work

Additional Supervision from Auxiliary Tasks When there is not enough data to learn a task, it is common to introduce additional supervision from some related auxiliary tasks. For example, in semi-supervised learning, previous work has employed various kinds of manually-designed supervision on unlabeled data [48, 56, 64]. In reinforcement leaning, due to sample inefficiency, auxiliary tasks (*e.g.* vision prediction [43], reward prediction [55]) are jointly trained to speed up convergence. In transfer learning or domain adaptation, models are trained on related domains/tasks and generalized to unseen domains [4, 8, 57]. Learning using privileged information (LUPI) also employs additional knowledge (*e.g.* meta data, additional modality) during training time [24, 54, 59]. However, LUPI does not emphasize the scarcity of training data as in our problem setting.

Multi-task Learning A highly related setting is multi-task learning (MTL). In MTL, models are trained to give high performance on different tasks simultaneously. Note that this is different from our setting because we only care about the performance on the main task. MTL is typically conducted through parameter sharing [2, 53], or prior sharing in a Bayesian manner [5, 23, 61, 62]. Parameter sharing and joint learning can achieve better generalization over learning each task independently [2], which also motivates our work. MTL has wide applications in areas including vision [7], language [12], speech [27], *etc*. We refer interested readers to this review [51].

Adaptive Task Reweighting When learning multiple tasks, it is important to estimate the relationship between different tasks in order to balance multiple losses. In MTL, this is usually realized by task clustering through a mixture prior [5, 17, 41, 65]. However, this type of methods only screens out unrelated tasks without further differentiating related tasks. Another line of work balances multiple losses based on gradient norm [11] or uncertainty [25, 31]. In our problem setting, the focus is changed to estimate the relationship between the main task and auxiliary tasks. In [63] task relationship is estimated based on whether the representation learned for one task can be easily reused for another task, which requires exhaustive enumeration of all the tasks. In [1], the enumeration process is vastly simplified by only considering a local landscape in the parameter space. However, a local landscape may be insufficient to represent the whole parameter distribution, especially in high dimensional cases such as deep networks. Recently, algorithms have been designed to adaptively reweight multiple tasks on the fly. For example, in [14] tasks are filtered out when having opposite gradient direction to the main task. The most similar work to ours is [37], where the task relationship is also estimated from similarity between gradients. However, unlike ours, they use inner product as similarity metric with the goal of speeding up training.

5 Conclusions

In this work, we develop an algorithm to automatically reweight auxiliary tasks, so that the data requirement for the main task is minimized. We first formulate the weighted likelihood function of auxiliary tasks as a surrogate prior for the main task. Then the optimal weights are obtained by minimizing the divergence between the surrogate prior and the true prior. We design a practical algorithm by turning the optimization problem into minimizing the distance between main task gradient and auxiliary task gradients. We demonstrate its effectiveness and robustness in reducing the data requirement under various settings including the extreme case of only a few examples.

Broader Impact

In this work we focus on using auxiliary tasks to compensate for data scarcity of a main task, and propose an algorithm to automatically reweight auxiliary tasks so that the data requirement on the main task is minimized. On the bright side, this could impact the industry and society from two aspects. First, this may help the machine learning techniques land in situations where labeled data is scarce or even unavailable, which is common in the real world. Second, our method can save the practitioners' time on manually tuning the weights, and also save the energy wasted for multiple runs, which is crucial considering that deep learning experiments nowadays are extremely eco-unfriendly.

However, our method may lead to negative consequences if it is not used right. For example, our method may need less data to extract information from a private dataset or system. Besides, our method may still fail in some situations, which may undermine the whole system if our method is over-trusted by practitioners.

References

[1] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6430–6439, 2019.

[2] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.

[3] Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. *arXiv preprint arXiv:1902.09229*, 2019.

[4] Nader Asadi, Mehrdad Hosseinzadeh, and Mahdi Eftekhari. Towards shape biased unsupervised representation learning for domain generalization. *arXiv preprint arXiv:1909.08245*, 2019.

[5] Bart Bakker and Tom Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4(May):83–99, 2003.

[6] Jonathan Baxter. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine learning*, 28(1):7–39, 1997.

[7] Hakan Bilen and Andrea Vedaldi. Integrated perception with recurrent multi-task neural networks. In *Advances in neural information processing systems*, pages 235–243, 2016.

[8] Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2229–2238, 2019.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[10] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[11] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *arXiv preprint arXiv:1711.02257*, 2017.

[12] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008.

[13] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[14] Yunshu Du, Wojciech M Czarnecki, Siddhant M Jayakumar, Razvan Pascanu, and Balaji Lakshminarayanan. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224*, 2018.

[15] Antonio D'Innocente and Barbara Caputo. Domain generalization with domain-specific aggregation modules. In *German Conference on Pattern Recognition*, pages 187–198. Springer, 2018.

[16] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Transactions on graphics (TOG)*, 31(4):1–10, 2012.

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

[18] Yanwei Fu, Timothy M Hospedales, Tao Xiang, and Shaogang Gong. Transductive multi-view zero-shot learning. *IEEE transactions on pattern analysis and machine intelligence*, 37(11):2332–2345, 2015.

[19] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536, 2005.

[20] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[21] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[23] TM Heskes. Empirical bayes for learning to learn. In *Proceedings of the 17th international conference on Machine learning*, pages 364–367, 2000.

[24] Judy Hoffman, Saurabh Gupta, and Trevor Darrell. Learning with side information through modality hallucination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 826–834, 2016.

[25] Hanzhang Hu, Debadeepta Dey, Martial Hebert, and J Andrew Bagnell. Learning anytime predictions in neural networks via adaptive loss balancing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3812–3821, 2019.

[26] Tianyang Hu, Zixiang Chen, Hanxi Sun, Jincheng Bai, Mao Ye, and Guang Cheng. Stein neural sampler. *arXiv preprint arXiv:1810.03545*, 2018.

[27] Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and Yifan Gong. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7304–7308. IEEE, 2013.

[28] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(Apr):695–709, 2005.

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

[30] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[31] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

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

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

[34] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

[35] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 2, 2013.

[36] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017.

[37] Xingyu Lin, Harjatin Baweja, George Kantor, and David Held. Adaptive auxiliary task weighting for reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4773–4784, 2019.

[38] Qiang Liu, Jason Lee, and Michael Jordan. A kernelized stein discrepancy for goodness-of-fit tests. In *International conference on machine learning*, pages 276–284, 2016.

[39] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in neural information processing systems*, pages 2378–2386, 2016.

[40] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[41] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and S Yu Philip. Learning multiple tasks with multilinear relationship networks. In *Advances in neural information processing systems*, pages 1594–1603, 2017.

[42] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[43] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[44] Saeid Motiian, Quinn Jones, Seyed Iranmanesh, and Gianfranco Doretto. Few-shot adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 6670–6680, 2017.

[45] Iain Murray and Zoubin Ghahramani. Bayesian learning in undirected graphical models: approximate mcmc algorithms. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 392–399. AUAI Press, 2004.

[46] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[47] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[48] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, pages 3235–3246, 2018.

[49] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[50] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[51] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[52] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016.

[53] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pages 527–538, 2018.

[54] Viktoriia Sharmanska, Novi Quadrianto, and Christoph H Lampert. Learning to rank using privileged information. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 825–832, 2013.

[55] Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307*, 2016.

[56] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.

[57] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4068–4076, 2015.

[58] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[59] Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5-6):544–557, 2009.

[60] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

[61] Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8(Jan):35–63, 2007.

[62] Kai Yu, Volker Tresp, and Anton Schwaighofer. Learning gaussian processes from multiple tasks. In *Proceedings of the 22nd international conference on Machine learning*, pages 1012–1019, 2005.

[63] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3712–3722, 2018.

[64] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE international conference on computer vision*, pages 1476–1485, 2019.

[65] Yu Zhang and Dit-Yan Yeung. A convex formulation for learning task relationships in multi-task learning. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 733–742, 2010.

A Additional Discussion on ARML

In this section we add more discussion on validity and soundness of ARML, especially on the three problems (**True Prior (P1)**, **Samples (P2)**, **Partition Function (P3)**), and how we resolve them (Sec. 3.3).

A.1 Full Version and Proof of Theorem 1 (P1)

In **True Prior (P1)** (Sec. 3.3) we use

$$\min_{\boldsymbol{\alpha}} E_{\theta \sim p^J} \log \frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)} \tag{12}$$

as a surrogate objective for the original optimization problem

$$\min_{\boldsymbol{\alpha}} D_{\mathrm{KL}}(p^*(\theta) \parallel p_{\boldsymbol{\alpha}}(\theta)). \tag{13}$$

In this section, we will first intuitively explain why optimizing (12) can end up with a near-optimal solution for (13), and what assumptions do we need to make. Then we will give the full version of Theorem 1 and also the proof.

Let $f(\boldsymbol{\alpha}) = E_{\theta\sim p^J} \log \frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)} = \frac{1}{Z(\boldsymbol{\alpha})} \int p^m(\theta)p_{\boldsymbol{\alpha}}(\theta) \log \frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)} d\theta$ be the optimization objective in (12), where $p^J(\theta) = \frac{p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)}{Z(\boldsymbol{\alpha})}$ and $Z(\boldsymbol{\alpha}) = \int p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)d\theta$ is the normalization term. Assume $p^*(\theta)$ has a compact support set S. Then we can write $f(\boldsymbol{\alpha})$ as

$$\begin{aligned}
f(\boldsymbol{\alpha}) &= \frac{1}{Z(\boldsymbol{\alpha})}\int_{\theta\in S} p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)\log\frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)}d\theta + \frac{1}{Z(\boldsymbol{\alpha})}\int_{\theta\notin S} p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)\log\frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)}d\theta \\
&= \frac{Z(S;\boldsymbol{\alpha})}{Z(S;\boldsymbol{\alpha}) + Z(\bar{S};\boldsymbol{\alpha})}\int_{\theta\in S}\frac{p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)}{Z(S;\boldsymbol{\alpha})}\log\frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)}d\theta \\
&\quad + \frac{Z(\bar{S};\boldsymbol{\alpha})}{Z(S;\boldsymbol{\alpha}) + Z(\bar{S};\boldsymbol{\alpha})}\int_{\theta\notin S}\frac{p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)}{Z(\bar{S};\boldsymbol{\alpha})}\log\frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)}d\theta \\
&= f(\boldsymbol{\alpha};S) + f(\boldsymbol{\alpha};\bar{S}),
\end{aligned} \tag{14}$$

where we denote the first and second term by $f(\boldsymbol{\alpha};S)$ and $f(\boldsymbol{\alpha};\bar{S})$ respectively, $Z(S;\boldsymbol{\alpha}) = \int_{\theta\in S} p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)d\theta$ and $Z(\bar{S};\boldsymbol{\alpha}) = \int_{\theta\notin S} p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)d\theta$ are the normalization terms inside and outside S.

To build the connection between the surrogate objective $f(\boldsymbol{\alpha})$ and the original objective $KL_{\boldsymbol{\alpha}} := D_{\mathrm{KL}}(p^*(\theta) \parallel p_{\boldsymbol{\alpha}}(\theta))$, we make the following assumption, The support set S is small so that $p_{\boldsymbol{\alpha}}(\theta)$ and $p^m(\theta)$ are constants inside S, and $p^*(\theta)$ is uniform in S. This assumption is reasonable when S is really informative, which we assume is the case for the true prior $p^*(\theta)$ [6]. With this assumption, we have

$$KL_{\boldsymbol{\alpha}} = \int_{\theta\in S} p^*(\theta)\log\frac{p^*(\theta)}{p_{\boldsymbol{\alpha}}(\theta)}d\theta = \log\frac{p^*(\theta^*)}{p_{\boldsymbol{\alpha}}(\theta^*)}\cdot\int_{\theta\in S}p^*(\theta)d\theta = \log\frac{p^*(\theta^*)}{p_{\boldsymbol{\alpha}}(\theta^*)}, \tag{15}$$

where $\theta^* \in S$ is the optimal parameter. We can also write $f(\boldsymbol{\alpha};S)$ as

$$\begin{aligned}
f(\boldsymbol{\alpha};S) &= \frac{Z(S;\boldsymbol{\alpha})}{Z(S;\boldsymbol{\alpha}) + Z(\bar{S};\boldsymbol{\alpha})}\int_{\theta\in S}\frac{p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)}{Z(S;\boldsymbol{\alpha})}\log\frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)}d\theta \\
&= \frac{Z(S;\boldsymbol{\alpha})}{Z(S;\boldsymbol{\alpha}) + Z(\bar{S};\boldsymbol{\alpha})}\log\frac{p^m(\theta^*)}{p_{\boldsymbol{\alpha}}(\theta^*)}\cdot\int_{\theta\in S}\frac{p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)}{Z(S;\boldsymbol{\alpha})}d\theta \\
&= \frac{Z(S;\boldsymbol{\alpha})}{Z(S;\boldsymbol{\alpha}) + Z(\bar{S};\boldsymbol{\alpha})}\log\frac{p^m(\theta^*)}{p_{\boldsymbol{\alpha}}(\theta^*)} \\
&= \frac{Z(S;\boldsymbol{\alpha})}{Z(S;\boldsymbol{\alpha}) + Z(\bar{S};\boldsymbol{\alpha})}(\log\frac{p^*(\theta^*)}{p_{\boldsymbol{\alpha}}(\theta^*)} + \log\frac{p^m(\theta^*)}{p^*(\theta^*)}) \\
&= \frac{Z(S;\boldsymbol{\alpha})}{Z(S;\boldsymbol{\alpha}) + Z(\bar{S};\boldsymbol{\alpha})}(KL_{\boldsymbol{\alpha}} + C_1),
\end{aligned} \tag{16}$$

where $C_1 = \log\frac{p^m(\theta^*)}{p^*(\theta^*)}$ is a constant invariant to $\boldsymbol{\alpha}$. Since $p^m(\theta)$ also covers other "overfitting" area other than S, we can assume that $p^*(\theta^*) \geq p^m(\theta^*)$, which gives $C_1 \leq 0$. Furthermore, we can notice that

$$Z(S;\boldsymbol{\alpha}) = \int_{\theta\in S}p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)d\theta = \int_{\theta\in S}\frac{p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)}{p^*(\theta)}p^*(\theta)d\theta = \frac{p^m(\theta^*)p_{\boldsymbol{\alpha}}(\theta^*)}{p^*(\theta^*)} = C_2e^{-KL_{\boldsymbol{\alpha}}}, \tag{17}$$

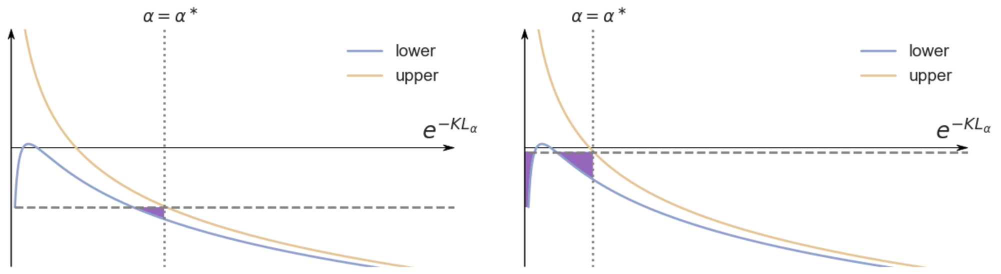
where $C_2 = p^m(\theta^*)$ is a constant invariant to $\boldsymbol{\alpha}$. Then we can write $f(\boldsymbol{\alpha};S)$ as

$$f(\boldsymbol{\alpha};S) = \frac{C_2e^{-KL_{\boldsymbol{\alpha}}}}{C_2e^{-KL_{\boldsymbol{\alpha}}} + Z(\bar{S};\boldsymbol{\alpha})}(KL_{\boldsymbol{\alpha}} + C_1). \tag{18}$$

In this way, we build the connection between the surrogate objective $f(\boldsymbol{\alpha})$ and the original objective $KL_{\boldsymbol{\alpha}}$.

Now we give an intuitive explanation for why optimizing $f(\boldsymbol{\alpha})$ gives a small $KL_{\boldsymbol{\alpha}}$ as well. We can write $f(\boldsymbol{\alpha})$ as

$$\begin{aligned}
f(\boldsymbol{\alpha}) &= f(\boldsymbol{\alpha};S) + f(\boldsymbol{\alpha};\bar{S}) \\
&= \frac{C_2e^{-KL_{\boldsymbol{\alpha}}}}{C_2e^{-KL_{\boldsymbol{\alpha}}} + Z(\bar{S};\boldsymbol{\alpha})}(KL_{\boldsymbol{\alpha}} + C_1) + \frac{Z(\bar{S};\boldsymbol{\alpha})}{C_2e^{-KL_{\boldsymbol{\alpha}}} + Z(\bar{S};\boldsymbol{\alpha})}\int_{\theta\in\bar{S}}\frac{p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)}{Z(\bar{S};\boldsymbol{\alpha})}\log\frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)}d\theta.
\end{aligned} \tag{19}$$



(a) $e^{-KL_{\boldsymbol{\alpha}^*}}$ is large. (b) $e^{-KL_{\boldsymbol{\alpha}^*}}$ is small.

Figure 5: $f(e^{-KL_{\boldsymbol{\alpha}}})$'s upper bound $f_u(e^{-KL_{\boldsymbol{\alpha}}})$ (golden line) and lower bound $f_l(e^{-KL_{\boldsymbol{\alpha}}})$ (blue line). $\boldsymbol{\alpha}^* =_{\boldsymbol{\alpha}} (e^{-KL_{\boldsymbol{\alpha}}}) =_{\boldsymbol{\alpha}} KL_{\boldsymbol{\alpha}}$ denotes the largest $e^{-KL_{\boldsymbol{\alpha}}}$ we could possibly reach. Shaded region denotes where $(e^{-KL_{\hat{\boldsymbol{\alpha}}}}, f(e^{-KL_{\hat{\boldsymbol{\alpha}}}}))$ could possibly be.

As one can notice, $f(\boldsymbol{\alpha})$ not only depends on $KL_{\boldsymbol{\alpha}}$, but also on $Z(\bar{S};\boldsymbol{\alpha})$ and the integral $\int_{\theta\in\bar{S}} \frac{p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)}{Z(\bar{S};\boldsymbol{\alpha})} \log \frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)} d\theta$. First we remove the dependency on the integral by taking its lower bound and upper bound. Concretely, with Jensen's inequality, we have

$$\int_{\theta\in\bar{S}} \frac{p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)}{Z(\bar{S};\boldsymbol{\alpha})} \log \frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)} d\theta \le \log \frac{\int_{\theta\in\bar{S}}(p^m(\theta))^2 d\theta}{Z(\bar{S};\boldsymbol{\alpha})} = \log \frac{C_3}{Z(\bar{S};\boldsymbol{\alpha})}, \tag{20}$$

where $C_3 = \int_{\theta\in\bar{S}}(p^m(\theta))^2 d\theta$ is a constant invariant to $\boldsymbol{\alpha}$. Likewise, we have

$$\begin{aligned}\int_{\theta\in\bar{S}} \frac{p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)}{Z(\bar{S};\boldsymbol{\alpha})} \log \frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)} d\theta &= \int_{\theta\in\bar{S}} -\frac{p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)}{Z(\bar{S};\boldsymbol{\alpha})} \log \frac{p_{\boldsymbol{\alpha}}(\theta)}{p^m(\theta)} d\theta \\ &\ge -\log \frac{\int_{\theta\in\bar{S}}(p_{\boldsymbol{\alpha}}(\theta))^2 d\theta}{Z(\bar{S};\boldsymbol{\alpha})} \\ &\ge -\log \frac{C_4}{Z(\bar{S};\boldsymbol{\alpha})},\end{aligned} \tag{21}$$

where $C_4 = \max_{\boldsymbol{\alpha}} \int_{\theta\in\bar{S}}(p_{\boldsymbol{\alpha}}(\theta))^2 d\theta$ is a constant invariant to $\boldsymbol{\alpha}$. In this way, we get the lower bound and upper bound for $f(\boldsymbol{\alpha})$:

$$\begin{aligned} f(\boldsymbol{\alpha}) \ge f_l(\boldsymbol{\alpha}) &= \frac{C_2 e^{-KL_{\boldsymbol{\alpha}}}}{C_2 e^{-KL_{\boldsymbol{\alpha}}} + Z(\bar{S};\boldsymbol{\alpha})}(KL_{\boldsymbol{\alpha}} + C_1) - \frac{Z(\bar{S};\boldsymbol{\alpha})}{C_2 e^{-KL_{\boldsymbol{\alpha}}} + Z(\bar{S};\boldsymbol{\alpha})} \log \frac{C_4}{Z(\bar{S};\boldsymbol{\alpha})}, \\ f(\boldsymbol{\alpha}) \le f_u(\boldsymbol{\alpha}) &= \frac{C_2 e^{-KL_{\boldsymbol{\alpha}}}}{C_2 e^{-KL_{\boldsymbol{\alpha}}} + Z(\bar{S};\boldsymbol{\alpha})}(KL_{\boldsymbol{\alpha}} + C_1) + \frac{Z(\bar{S};\boldsymbol{\alpha})}{C_2 e^{-KL_{\boldsymbol{\alpha}}} + Z(\bar{S};\boldsymbol{\alpha})} \log \frac{C_3}{Z(\bar{S};\boldsymbol{\alpha})}. \end{aligned} \tag{22}$$

We plot f_l and f_u as functions of $e^{-KL_{\boldsymbol{\alpha}}}$ in Fig. 5 (here we assume $Z(\bar{S};\boldsymbol{\alpha})$ is constant w.r.t. $\boldsymbol{\alpha}$ for brevity). $f(\boldsymbol{\alpha})$ lies between the upper bound (golden line) and the lower bound (blue line).

Our goal is to find the optimal $\boldsymbol{\alpha}^*$ that minimizes $KL_{\boldsymbol{\alpha}}$, *i.e.*, $\boldsymbol{\alpha}^* =_{\boldsymbol{\alpha}} KL_{\boldsymbol{\alpha}} =_{\boldsymbol{\alpha}} e^{-KL_{\boldsymbol{\alpha}}}$. By optimizing $f(\boldsymbol{\alpha})$, we end up with a suboptimal $\hat{\boldsymbol{\alpha}} =_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha})$. Ideally, we hope that $KL_{\hat{\boldsymbol{\alpha}}}$ is close to $KL_{\boldsymbol{\alpha}^*}$, which means when we minimize $f(\hat{\boldsymbol{\alpha}})$, we can also get a large $e^{-KL_{\hat{\boldsymbol{\alpha}}}}$. This is the case when $e^{-KL_{\boldsymbol{\alpha}^*}}$ is large (see Fig. 5a). When $e^{-KL_{\boldsymbol{\alpha}^*}}$ is large, the upper bound f_u and the lower bound f_l are close to each other around $e^{-KL_{\boldsymbol{\alpha}^*}}$ (this is the case when $Z(\bar{S};\boldsymbol{\alpha})$ is small). Since we have

$$f_l(e^{-KL_{\hat{\boldsymbol{\alpha}}}}) \le f(e^{-KL_{\hat{\boldsymbol{\alpha}}}}) \le f(e^{-KL_{\boldsymbol{\alpha}^*}}) \le f_u(e^{-KL_{\boldsymbol{\alpha}^*}}), \tag{23}$$

we can assert that $(e^{-KL_{\hat{\boldsymbol{\alpha}}}}, f(e^{-KL_{\hat{\boldsymbol{\alpha}}}}))$ lies in the shaded region, because if $e^{-KL_{\hat{\boldsymbol{\alpha}}}}$ is on the left side of the region, we have $f(e^{-KL_{\hat{\boldsymbol{\alpha}}}}) \ge f_u(e^{-KL_{\boldsymbol{\alpha}^*}})$ which is contradictary to (23), and if $e^{-KL_{\hat{\boldsymbol{\alpha}}}}$ cannot be on the right side of the region because $e^{-KL_{\boldsymbol{\alpha}^*}}$ is the furthest we can go. Since the shaded region is small, $KL_{\hat{\boldsymbol{\alpha}}}$ is thus close to the optimal solution $KL_{\boldsymbol{\alpha}^*}$.

Unfortunately, this may not hold anymore when $e^{-KL_{\boldsymbol{\alpha}^*}}$ is small (see Fig. 5b). This is because f_l will reach a local minima when $e^{-KL_{\boldsymbol{\alpha}}} \to 0$. If $e^{-KL_{\boldsymbol{\alpha}^*}}$ is not large enough, it may be higher than $\lim_{e^{-KL_{\boldsymbol{\alpha}}}\to 0} f_l(e^{-KL_{\boldsymbol{\alpha}}})$, which means the shaded region near y-axis is also included. In this region $f(\boldsymbol{\alpha})$ could be really small (which is the goal when optimizing the surrogate objective $f(\boldsymbol{\alpha})$), but $KL_{\boldsymbol{\alpha}}$ could be extremely large.

To avoid this situation, we only have to assume that

$$f_u(e^{-KL_{\boldsymbol{\alpha}^*}}) \leq \lim_{e^{-KL_{\boldsymbol{\alpha}}} \to 0} f_l(e^{-KL_{\boldsymbol{\alpha}}}) = -\log \frac{C_4}{Z(\bar{S};\boldsymbol{\alpha})}, \tag{24}$$

or if we denote $\gamma_1 = \min_{\boldsymbol{\alpha}} Z(\bar{S};\boldsymbol{\alpha})$ and $\gamma_2 = \max_{\boldsymbol{\alpha}} Z(\bar{S};\boldsymbol{\alpha})$, then we only need the following assumption: The optimal $KL_{\boldsymbol{\alpha}^*}$ is small so that $f_u(e^{-KL_{\boldsymbol{\alpha}^*}}) \leq -\log \frac{C_4}{\gamma_1}$. This assumption holds as long as there is at least one task that is related to the main task (having a small $KL_{\boldsymbol{\alpha}}$), which is reasonable because if all the tasks are unrelated, then reweighing is also meaningless. See the remark below for more discussion on the validity of the assumption.

Now we give the formal version of the theorem:

Theorem 2. *(formal version) With Assumption 1, 2, if* $\gamma_2 \leq \min(\frac{C_3}{e}, \frac{C_4}{e})$*, then we have*

$$KL_{\hat{\boldsymbol{\alpha}}} \leq KL_{\boldsymbol{\alpha}^*} + \frac{2\gamma_2^2}{C} \log \frac{C'}{\gamma_2} \tag{25}$$

Proof. From Assumption A.1 we have

$$\frac{C_2 e^{-KL_{\boldsymbol{\alpha}^*}}}{C_2 e^{-KL_{\boldsymbol{\alpha}^*}} + Z(\bar{S};\boldsymbol{\alpha}^*)}(KL_{\boldsymbol{\alpha}^*} + C_1) + \frac{Z(\bar{S};\boldsymbol{\alpha}^*)}{C_2 e^{-KL_{\boldsymbol{\alpha}^*}} + Z(\bar{S};\boldsymbol{\alpha}^*)} \log \frac{C_3}{Z(\bar{S};\boldsymbol{\alpha}^*)} \leq -\log \frac{C_4}{\gamma_1}. \tag{26}$$

Since $\gamma_2 \leq C_3$ and $\gamma_2 \leq C_4$, we have $\log \frac{C_3}{Z(\bar{S};\boldsymbol{\alpha}^*)} \geq \log \frac{C_3}{\gamma_2} \geq 0$, and $-\log \frac{C_4}{\gamma_1} \leq -\log \frac{C_4}{\gamma_2} \leq 0$. Then leaves us $KL_{\boldsymbol{\alpha}^*} + C_1 \leq 0$ in order to make (26) satisfied. Then we can relax (26) into

$$KL_{\boldsymbol{\alpha}^*} + C_1 \leq -\log \frac{C_4}{\gamma_1}, \tag{27}$$

which gives

$$C_2 e^{-KL_{\boldsymbol{\alpha}^*}} \geq \frac{C_5}{\gamma_1}, \tag{28}$$

where $C_5 = C_4 C_2 e^{C_1}$. This bounds the value of $KL_{\boldsymbol{\alpha}^*}$.

Moreover, from (23) and Assumption A.1 we have

$$f_l(e^{-KL_{\hat{\boldsymbol{\alpha}}}}) \leq f_u(e^{-KL_{\boldsymbol{\alpha}^*}}) \leq -\log \frac{C_4}{\gamma_1}, \tag{29}$$

which gives

$$f_l(e^{-KL_{\hat{\boldsymbol{\alpha}}}}) = \frac{C_2 e^{-KL_{\hat{\boldsymbol{\alpha}}}}}{C_2 e^{-KL_{\hat{\boldsymbol{\alpha}}}} + Z(\bar{S};\hat{\boldsymbol{\alpha}})}(KL_{\hat{\boldsymbol{\alpha}}} + C_1) - \frac{Z(\bar{S};\hat{\boldsymbol{\alpha}})}{C_2 e^{-KL_{\hat{\boldsymbol{\alpha}}}} + Z(\bar{S};\hat{\boldsymbol{\alpha}})} \log \frac{C_4}{Z(\bar{S};\hat{\boldsymbol{\alpha}})} \leq -\log \frac{C_4}{\gamma_1}. \tag{30}$$

Since $Z(\bar{S};\hat{\boldsymbol{\alpha}}) \geq \gamma_1$, we can relax (30) into

$$\frac{C_2 e^{-KL_{\hat{\boldsymbol{\alpha}}}}}{C_2 e^{-KL_{\hat{\boldsymbol{\alpha}}}} + Z(\bar{S};\hat{\boldsymbol{\alpha}})}(KL_{\hat{\boldsymbol{\alpha}}} + C_1) - \frac{Z(\bar{S};\hat{\boldsymbol{\alpha}})}{C_2 e^{-KL_{\hat{\boldsymbol{\alpha}}}} + Z(\bar{S};\hat{\boldsymbol{\alpha}})} \log \frac{C_4}{Z(\bar{S};\hat{\boldsymbol{\alpha}})} \leq -\log \frac{C_4}{Z(\bar{S};\hat{\boldsymbol{\alpha}})}, \tag{31}$$

which can be simplified into

$$KL_{\hat{\boldsymbol{\alpha}}} + C_1 \leq -\log \frac{C_4}{Z(\bar{S};\hat{\boldsymbol{\alpha}})} \leq -\log \frac{C_4}{\gamma_2}, \tag{32}$$

which means

$$C_2 e^{-KL_{\hat{\boldsymbol{\alpha}}}} \geq \frac{C_5}{\gamma_2}. \tag{33}$$

This bounds the value of $KL_{\hat{\boldsymbol{\alpha}}}$.

Now we build the connection between $KL_{\hat{\boldsymbol{\alpha}}}$ and $KL_{\boldsymbol{\alpha}^*}$. Since $f_l(e^{-KL_{\hat{\boldsymbol{\alpha}}}}) \leq f_u(e^{-KL_{\boldsymbol{\alpha}^*}})$, we have

$$\begin{aligned} &\frac{C_2 e^{-KL_{\hat{\boldsymbol{\alpha}}}}}{C_2 e^{-KL_{\hat{\boldsymbol{\alpha}}}} + Z(\bar{S};\hat{\boldsymbol{\alpha}})}(KL_{\hat{\boldsymbol{\alpha}}} + C_1) - \frac{Z(\bar{S};\hat{\boldsymbol{\alpha}})}{C_2 e^{-KL_{\hat{\boldsymbol{\alpha}}}} + Z(\bar{S};\hat{\boldsymbol{\alpha}})} \log \frac{C_4}{Z(\bar{S};\hat{\boldsymbol{\alpha}})} \\ \leq &\frac{C_2 e^{-KL_{\boldsymbol{\alpha}^*}}}{C_2 e^{-KL_{\boldsymbol{\alpha}^*}} + Z(\bar{S};\boldsymbol{\alpha}^*)}(KL_{\boldsymbol{\alpha}^*} + C_1) + \frac{Z(\bar{S};\boldsymbol{\alpha}^*)}{C_2 e^{-KL_{\boldsymbol{\alpha}^*}} + Z(\bar{S};\boldsymbol{\alpha}^*)} \log \frac{C_3}{Z(\bar{S};\boldsymbol{\alpha}^*)}. \end{aligned} \tag{34}$$

Since $KL_{\hat{\boldsymbol{\alpha}}} + C_1 \leq -\log \frac{C_4}{\gamma_2} \leq 0$, $KL_{\boldsymbol{\alpha}^*} \geq 0$, and also with (28) and (33), we can relax (34) into

$$\begin{aligned} & KL_{\hat{\boldsymbol{\alpha}}} + C_1 - \frac{Z(\bar{S};\hat{\boldsymbol{\alpha}})}{C_5/\gamma_2} \log \frac{C_4}{Z(\bar{S};\hat{\boldsymbol{\alpha}})} \\ \leq\ & KL_{\boldsymbol{\alpha}^*} + \frac{C_2 e^{-KL_{\boldsymbol{\alpha}^*}}}{C_2 e^{-KL_{\boldsymbol{\alpha}^*}} + Z(\bar{S};\boldsymbol{\alpha}^*)} C_1 + \frac{Z(\bar{S};\boldsymbol{\alpha}^*)}{C_5/\gamma_1} \log \frac{C_3}{Z(\bar{S};\boldsymbol{\alpha}^*)}, \end{aligned} \tag{35}$$

which gives

$$KL_{\hat{\boldsymbol{\alpha}}} \leq KL_{\boldsymbol{\alpha}^*} - \frac{Z(\bar{S};\boldsymbol{\alpha}^*)}{C_2 e^{-KL_{\boldsymbol{\alpha}^*}} + Z(\bar{S};\boldsymbol{\alpha}^*)} C_1 + \frac{Z(\bar{S};\hat{\boldsymbol{\alpha}})}{C_5/\gamma_2} \log \frac{C_4}{Z(\bar{S};\hat{\boldsymbol{\alpha}})} + \frac{Z(\bar{S};\boldsymbol{\alpha}^*)}{C_5/\gamma_1} \log \frac{C_3}{Z(\bar{S};\boldsymbol{\alpha}^*)}. \tag{36}$$

Since $Z(\bar{S};\hat{\boldsymbol{\alpha}}) \leq \gamma_2 \leq \frac{C_4}{e}$, we have $Z(\bar{S};\hat{\boldsymbol{\alpha}}) \log \frac{C_4}{Z(\bar{S};\hat{\boldsymbol{\alpha}})} \leq \gamma_2 \log \frac{C_4}{\gamma_2}$. Similarly, we have $Z(\bar{S};\boldsymbol{\alpha}^*) \log \frac{C_3}{Z(\bar{S};\boldsymbol{\alpha}^*)} \leq \gamma_2 \log \frac{C_3}{\gamma_2}$. Then we have

$$KL_{\hat{\boldsymbol{\alpha}}} \leq KL_{\boldsymbol{\alpha}^*} - \frac{Z(\bar{S};\boldsymbol{\alpha}^*)}{C_2 e^{-KL_{\boldsymbol{\alpha}^*}} + Z(\bar{S};\boldsymbol{\alpha}^*)} C_1 + \frac{\gamma_2^2}{C_5} \log \frac{C_4}{\gamma_2} + \frac{\gamma_2^2}{C_5} \log \frac{C_3}{\gamma_2}. \tag{37}$$

Since $C_1 \leq 0$, we can get

$$KL_{\hat{\boldsymbol{\alpha}}} \leq KL_{\boldsymbol{\alpha}^*} + \frac{\gamma_2^2}{C_5}(-C_1) + \frac{\gamma_2^2}{C_5} \log \frac{C_4}{\gamma_2} + \frac{\gamma_2^2}{C_5} \log \frac{C_3}{\gamma_2}, \tag{38}$$

which gives

$$KL_{\hat{\boldsymbol{\alpha}}} \leq KL_{\boldsymbol{\alpha}^*} + \frac{2\gamma_2^2}{C_5} \log \frac{C_6}{\gamma_2}, \tag{39}$$

where $C_6 = \sqrt{C_3 C_4 e^{-C_1}}$.

□

Remark. *From Theorem 2 we see that $KL_{\hat{\boldsymbol{\alpha}}}$ is close to $KL_{\boldsymbol{\alpha}^*}$ as long as γ_2 is small. One may notice that γ_2 cannot be arbitrarily small because from* (33) *we have*

$$\frac{C_5}{\gamma_2} \leq C_2 e^{-KL_{\hat{\boldsymbol{\alpha}}}} \leq C_2, \tag{40}$$

which means

$$\gamma_2 \geq \frac{C_5}{C_2} = C_4 e^{C_1}. \tag{41}$$

However, we can safely assume that

$$C_1 = \log \frac{p^m(\theta^*)}{p^*(\theta^*)} \ll 0 \tag{42}$$

since p^* is much more informative than p^m, especially when labeled data for the main task is scarce. This means γ_2 can be extremely small as long as C_1 is small, which makes $KL_{\hat{\boldsymbol{\alpha}}}$ close to $KL_{\boldsymbol{\alpha}^*}$. Similarly, Assumption A.1 can easily hold as long as C_1 is small.

A.2 Sampling through Langevin Dynamics (P2)

In **Samples (P2)** we use Langevin dynamics [46, 60] to sample from the distribution p^J. Concretely, at each iteration, we update θ by

$$\theta_{t+1} = \theta_t - \epsilon_t \nabla \mathcal{L}(\theta_t) + \eta_t, \tag{43}$$

where $\mathcal{L}(\theta) \propto -\log p^J(\theta)$ is the joint loss, and $\eta_t \sim N(0, 2\epsilon_t)$ is a Gaussian noise. In this way, θ_t converges to samples from p^J, which can be used to estimate our optimization objective. However, since we normally use a mini-batch estimator $\hat{\mathcal{L}}(\theta)$ to approximate $\mathcal{L}(\theta)$, this may introduce additional noise other than η_t, which may make the sampling procedure inaccurate. In [60] it is proposed to anneal the learning rate to zero so that the gradient stochasticity is dominated by the injected noise, thus alleviating the impact of mini-batch estimator. However we find in practice that the gradient noise is negligible compared to the injected noise (Table 5). Therefore, we ignore the gradient noise and directly inject the noise η_t into the updating step.

Table 5: Standard deviation of different types of noise. We find that the gradient noise is negligible compared to the injected noise.

	Standard deviation
Gradient Noise	$\sim 10^{-6}$
Injected Noise	$\sim 10^{-3}$

A.3 Score Function and Fisher Divergence (P3)

In **Partition Function (P3)** we propose to minimize

$$\min_{\boldsymbol{\alpha}} E_{\theta\sim p^J} \|\nabla \log p(\mathcal{T}_m|\theta) - \nabla \log p_{\boldsymbol{\alpha}}(\theta)\|_2^2 \tag{44}$$

as our final objective. Notice that

$$\begin{aligned}
&\min_{\boldsymbol{\alpha}} E_{\theta\sim p^J} \|\nabla \log p(\mathcal{T}_m|\theta) - \nabla \log p_{\boldsymbol{\alpha}}(\theta)\|_2^2 \\
\Leftrightarrow\ &\min_{\boldsymbol{\alpha}} E_{\theta\sim p^J} \|\nabla \log p^m(\theta) - \nabla \log p_{\boldsymbol{\alpha}}(\theta)\|_2^2 \\
\Leftrightarrow\ &\min_{\boldsymbol{\alpha}} E_{\theta\sim p^J} \|\nabla \log (p^m(\theta)\cdot p_{\boldsymbol{\alpha}}(\theta)) - 2\cdot\nabla \log p_{\boldsymbol{\alpha}}(\theta)\|_2^2 \\
\Leftrightarrow\ &\min_{\boldsymbol{\alpha}} E_{\theta\sim p^J} \|\nabla \log p^J(\theta) - \nabla \log p^2_{\boldsymbol{\alpha}}(\theta)\|_2^2 \\
\Leftrightarrow\ &\min_{\boldsymbol{\alpha}} F(p^J(\theta) \parallel \frac{1}{Z'(\boldsymbol{\alpha})} p^2_{\boldsymbol{\alpha}}(\theta)),
\end{aligned} \tag{45}$$

where $F(p(\theta) \parallel q(\theta)) = E_{\theta\sim p}\|\nabla \log p(\theta) - \nabla \log q(\theta)\|_2^2$ is the *Fisher divergence*, and $Z'(\boldsymbol{\alpha}) = \int p^2_{\boldsymbol{\alpha}}(\theta)d\theta$ is the normalization term. This means, by optimizing (44), we are actually minimizing the Fisher divergence between $p^J(\theta)$ and $\frac{1}{Z'(\boldsymbol{\alpha})}p^2_{\boldsymbol{\alpha}}(\theta)$. As pointed by [26, 38], Fisher divergence is stronger than KL divergence, which means by minimizing $F(p^J(\theta) \parallel \frac{1}{Z'(\boldsymbol{\alpha})}p^2_{\boldsymbol{\alpha}}(\theta))$, the KL divergence $D_{KL}(p^J(\theta) \parallel \frac{1}{Z'(\boldsymbol{\alpha})}p^2_{\boldsymbol{\alpha}}(\theta))$ is also bounded near the optimum up to a small error.

Therefore, optimizing (44) is equivalent to minimizing $D_{KL}(p^J(\theta) \parallel \frac{1}{Z'(\boldsymbol{\alpha})}p^2_{\boldsymbol{\alpha}}(\theta))$. Notice that

$$\begin{aligned}
&\min_{\boldsymbol{\alpha}} D_{KL}(p^J(\theta) \parallel \frac{1}{Z'(\boldsymbol{\alpha})}p^2_{\boldsymbol{\alpha}}(\theta)) \\
\Leftrightarrow\ &\min_{\boldsymbol{\alpha}} \int p^J(\theta) \log \frac{p^J(\theta)}{\frac{1}{Z'(\boldsymbol{\alpha})}p^2_{\boldsymbol{\alpha}}(\theta)} d\theta \\
\Leftrightarrow\ &\min_{\boldsymbol{\alpha}} \int p^J(\theta) \log \frac{\frac{1}{Z(\boldsymbol{\alpha})}p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)}{\frac{1}{Z'(\boldsymbol{\alpha})}p^2_{\boldsymbol{\alpha}}(\theta)} d\theta \\
\Leftrightarrow\ &\min_{\boldsymbol{\alpha}} \int p^J(\theta) \log \frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)} d\theta + \log \frac{Z'(\boldsymbol{\alpha})}{Z(\boldsymbol{\alpha})} \\
\Leftrightarrow\ &\min_{\boldsymbol{\alpha}} \int p^J(\theta) \log \frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)} d\theta + \log \frac{\int p^2_{\boldsymbol{\alpha}}(\theta)d\theta}{\int p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)d\theta}
\end{aligned} \tag{46}$$

is different from (12) only on the $\log \frac{\int p^2_{\boldsymbol{\alpha}}(\theta)d\theta}{\int p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)d\theta}$ term. To analyze the impact of this additional term, we assume that the likelihood function of each auxiliary task is a Gaussian, *i.e.*, $p(\mathcal{T}_{a_k}|\theta) \propto N(\theta|\theta_k, \mathbf{\Sigma})$, with mean θ_k and covariance $\mathbf{\Sigma}$. Then we have $p_{\boldsymbol{\alpha}}(\theta) = N(\theta|\sum_k \alpha_k\theta_k/K, \mathbf{\Sigma}/K)$ (note that $\sum_k \alpha_k = K$). In this case $\int p^2_{\boldsymbol{\alpha}}(\theta)d\theta$ only depends on $\mathbf{\Sigma}$ and is invariant to $\boldsymbol{\alpha}$. Thus optimizing (44) is equivalent to

$$\begin{aligned}
&\min_{\boldsymbol{\alpha}} D_{KL}(p^J(\theta) \parallel \frac{1}{Z'(\boldsymbol{\alpha})}p^2_{\boldsymbol{\alpha}}(\theta)) \\
\Leftrightarrow\ &\min_{\boldsymbol{\alpha}} \int p^J(\theta) \log \frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)} d\theta + \log \frac{\int p^2_{\boldsymbol{\alpha}}(\theta)d\theta}{\int p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)d\theta} \\
\Leftrightarrow\ &\min_{\boldsymbol{\alpha}} \int p^J(\theta) \log \frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)} d\theta - \log \int p^m(\theta)p_{\boldsymbol{\alpha}}(\theta)d\theta.
\end{aligned} \tag{47}$$

Denote the optimal solution for (47) by $\boldsymbol{\alpha}^\dagger$. Then we can build the connection between $\boldsymbol{\alpha}^\dagger$ and $\hat{\boldsymbol{\alpha}}$ by

$$\int p^J(\theta)\log\frac{p^m(\theta)}{p_{\boldsymbol{\alpha}^\dagger}(\theta)}d\theta - \log\int p^m(\theta)p_{\boldsymbol{\alpha}^\dagger}(\theta)d\theta \le \int p^J(\theta)\log\frac{p^m(\theta)}{p_{\hat{\boldsymbol{\alpha}}}(\theta)}d\theta - \log\int p^m(\theta)p_{\hat{\boldsymbol{\alpha}}}(\theta)d\theta. \quad (48)$$

Since $\hat{\boldsymbol{\alpha}}$ minimizes $\int p^J(\theta)\log\frac{p^m(\theta)}{p_{\boldsymbol{\alpha}}(\theta)}d\theta$, which means $\int p^J(\theta)\log\frac{p^m(\theta)}{p_{\hat{\boldsymbol{\alpha}}}(\theta)}d\theta \le \int p^J(\theta)\log\frac{p^m(\theta)}{p_{\boldsymbol{\alpha}^\dagger}(\theta)}d\theta$, we can get

$$-\log\int p^m(\theta)p_{\boldsymbol{\alpha}^\dagger}(\theta)d\theta \le -\log\int p^m(\theta)p_{\hat{\boldsymbol{\alpha}}}(\theta)d\theta, \quad (49)$$

or

$$\int p^m(\theta)p_{\boldsymbol{\alpha}^\dagger}(\theta)d\theta \ge \int p^m(\theta)p_{\hat{\boldsymbol{\alpha}}}(\theta)d\theta, \quad (50)$$

which gives

$$\int_{\theta\in S} p^m(\theta)p_{\boldsymbol{\alpha}^\dagger}(\theta)d\theta + \int_{\theta\in\bar{S}} p^m(\theta)p_{\boldsymbol{\alpha}^\dagger}(\theta)d\theta \ge \int_{\theta\in S} p^m(\theta)p_{\hat{\boldsymbol{\alpha}}}(\theta)d\theta + \int_{\theta\in\bar{S}} p^m(\theta)p_{\hat{\boldsymbol{\alpha}}}(\theta)d\theta. \quad (51)$$

Then we have

$$\begin{aligned}\int_{\theta\in S} p^m(\theta)p_{\boldsymbol{\alpha}^\dagger}(\theta)d\theta &\ge \int_{\theta\in S} p^m(\theta)p_{\hat{\boldsymbol{\alpha}}}(\theta)d\theta + \int_{\theta\in\bar{S}} p^m(\theta)p_{\hat{\boldsymbol{\alpha}}}(\theta)d\theta - \int_{\theta\in\bar{S}} p^m(\theta)p_{\boldsymbol{\alpha}^\dagger}(\theta)d\theta \\ &\ge \int_{\theta\in S} p^m(\theta)p_{\hat{\boldsymbol{\alpha}}}(\theta)d\theta - (\gamma_2-\gamma_1).\end{aligned} \quad (52)$$

From Assumption A.1 we have

$$\frac{p^m(\theta^*)p_{\boldsymbol{\alpha}^\dagger}(\theta^*)}{p^*(\theta^*)} \ge \frac{p^m(\theta^*)p_{\hat{\boldsymbol{\alpha}}}(\theta^*)}{p^*(\theta^*)} - (\gamma_2-\gamma_1), \quad (53)$$

which gives

$$KL_{\boldsymbol{\alpha}^\dagger} = -\log\frac{p_{\boldsymbol{\alpha}^\dagger}(\theta^*)}{p^*(\theta^*)} \le -\log\left(\frac{p_{\hat{\boldsymbol{\alpha}}}(\theta^*)}{p^*(\theta^*)} - \frac{\gamma_2-\gamma_1}{p^m(\theta^*)}\right) \le -\log\frac{p_{\hat{\boldsymbol{\alpha}}}(\theta^*)}{p^*(\theta^*)} + \frac{\gamma_2-\gamma_1}{p^m(\theta^*)}, \quad (54)$$

or

$$KL_{\boldsymbol{\alpha}^\dagger} \le KL_{\hat{\boldsymbol{\alpha}}} + \frac{\gamma_2}{C_2}. \quad (55)$$

After combining with Theorem 2, we have

$$KL_{\boldsymbol{\alpha}^\dagger} \le KL_{\boldsymbol{\alpha}^*} + \frac{2\gamma_2^2}{C_5}\log\frac{C_6}{\gamma_2} + \frac{\gamma_2}{C_2}. \quad (56)$$

This means by optimizing our final objective (44), the KL divergence $KL_{\boldsymbol{\alpha}^\dagger}$ is also bounded near the optimal value, which provides a theoretical justification of our algorithm.

A.4 Tips for Practitioners

In Section 2.4, we propose a two-stage algorithm, where we update the task weights with Langevin dynamics in the first stage, and then udpate the model with fixed task weights in the second stage. However, we find in practice that we can also find the similar task weights if we turn off the Langevin dynamics and directly sample from regular SGD. Therefore, we can further simplify the algorithm by removing the Langevin dynamics and merge the two stage, *i.e.*, update task weights and model parameters at the same time until convergence. This simplified version is summarized in Algorithm 2.

Algorithm 2 ARML (simplified version)

```
  Input: main task data $\mathcal{T}_m$, auxiliary task data $\mathcal{T}_{a_k}$, initial parameter $\theta_0$, initial task weights $\boldsymbol{\alpha}$
  Parameters: learning rate of $t$-th iteration $\epsilon_t$, learning rate for task weights $\beta$

  for iteration $t = 1$ to $T$ do
    $\theta_t \leftarrow \theta_{t-1} - \epsilon_t(-\nabla\log p(\mathcal{T}_m|\theta_{t-1}) - \sum_{k=1}^K \alpha_k\nabla\log p(\mathcal{T}_{a_k}|\theta_{t-1})) + \eta_t$
    $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \beta\nabla_{\boldsymbol{\alpha}}\|\nabla\log p(\mathcal{T}_m|\theta_t) - \sum_{k=1}^K \alpha_k\nabla\log p(\mathcal{T}_{a_k}|\theta_t)\|_2^2$
    Project $\boldsymbol{\alpha}$ back into $\mathcal{A}$
  end for
```

B Experimental Settings

For all results, we repeat experiments for three times and report the average performance. Error bars are reported with CI=95%. In our algorithm, the only hyperparameter is the learning rate β of task weights. Specifically, we find the results insensitive to the choice of β. Therefore, we randomly choose $\beta \in [0.0005, 0.05]$, for a trade-off between steady training and fast convergence. We use PyTorch [50] for implementation.

B.1 Semi-supervised Learning

For semi-supervised learning, we use two datasets, CIFAR10 [33] and SVHN [47]. For CIFAR10, we follow the standard train/validation split, with 45000 images for training and 5000 for validation. Only 4000 out of 45000 training images are labeled. For SVHN, we use the standard train/validation split with 65932 images for training and 7325 for validation. Only 1000 out of 65392 images are labeled. Both datasets can be downloaded from the official PyTorch torchvision library (`https://pytorch.org/docs/stable/torchvision/index.html`). Following [48], we use WRN-28-2 as our backbone, *i.e.*, ResNet [22] with depth 28 and width 2, including batch normalization [29] and leaky ReLU [42]. We train our model for 200000 iterations, using Adam [32] optimizer with batch size of 256 and learning rate of 0.005 in first 160000 iterations and 0.001 for the rest iterations.

For implementation of self-supervised semi-supervised learning (S4L), we follow the settings in the original paper [64]. Note that we make two differences from [64]: (i) for steadier training, we use the model with time-averaged parameters [56] to extract feature of the original image, (ii) To avoid over-sampling of negative samples in triplet-loss [3], we only put a loss on the cosine similarity between original feature and augmented feature.

B.2 Multi-label Classification

For multi-label classification, we use CelebA [40] as our dataset. It contains 200K face images, each labeled with 40 binary attributes. We cast this into a multi-label classification problem, where we randomly choose one attribute as the main classification task, and other 39 as auxiliary tasks. We randomly choose 1% images as labeled images for main task. The dataset is available at `http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html`. We use ResNet18 [22] as our backbone. We train the model for 90 epochs using SGD solver with batch size of 256 and scheduled learning rate of 0.1 initially and $0.1\times$ shrinked every 30 epochs.

B.3 Domain Generalization

Following the literature [4, 8], we use PACS [36] as our dataset for domain generalization. PACS consists of four domains (photo, art painting, cartoon and sketch), each containing 7 categories (dog, elephant, giraffe, guitar, horse, house and person). The dataset is created by intersecting classes in Caltech-256 [20], Sketchy [52], TU-Berlin [16] and Google Images. Dataset can be downloaded from `http://sketchx.eecs.qmul.ac.uk/`. Following protocol in [36], we split the images from training domains to 9 (train) : 1 (val) and test on the whole target domain. We use a simple data augmentation protocol by randomly cropping the images to 80-100% of original sizes and randomly apply horizontal flipping. We use ResNet18 [22] as our backbone. Models are trained with SGD solver, 100 epochs, batch size 128. Learning rate is set to 0.001 and shrinked down to 0.0001 after 80 epochs.