

# META-LEARNING WITH IMPLICIT PROCESSES

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

This paper presents a novel *implicit process-based meta-learning* (IPML) algorithm that, in contrast to existing works, explicitly represents each task as a continuous latent vector and models its probabilistic belief within the highly expressive IP framework. Unfortunately, meta-training in IPML is computationally challenging due to its need to perform intractable exact IP inference in task adaptation. To resolve this, we propose a novel expectation-maximization algorithm based on the stochastic gradient Hamiltonian Monte Carlo sampling method to perform meta-training. Our delicate design of the neural network architecture for meta-training in IPML allows competitive meta-learning performance to be achieved. Unlike existing works, IPML offers the benefits of being amenable to the characterization of a principled distance measure between tasks using the maximum mean discrepancy, active task selection without needing the assumption of known task contexts, and synthetic task generation by modeling task-dependent input distributions. Empirical evaluation on benchmark datasets shows that IPML outperforms existing Bayesian meta-learning algorithms. We have also empirically demonstrated on an e-commerce company’s real-world dataset that IPML outperforms the baselines and identifies “outlier” tasks which can potentially degrade meta-testing performance.

## 1 INTRODUCTION

Few-shot learning (also known as *meta-learning*) is a defining characteristic of human intelligence. Its goal is to leverage the experiences from previous tasks to form a model (represented by meta-parameters) that can rapidly adapt to a new task using only a limited quantity of its training data. A number of meta-learning algorithms (Finn et al., 2018; Jerfel et al., 2019; Ravi & Beato, 2018; Rusu et al., 2019; Yoon et al., 2018) have recently adopted a probabilistic perspective to characterize the uncertainty in the predictions via a Bayesian treatment of the meta-parameters. Though they can consequently represent different tasks with different values of meta-parameters, it is not clear how or whether they are naturally amenable to (a) the characterization of a principled similarity/distance measure between tasks (e.g., for identifying outlier tasks that can potentially hurt training for the new task, procuring the most valuable/similar tasks/datasets to the new task, detecting task distribution shift, among others), (b) active task selection given a limited budget of expensive task queries (see Appendix A.2.3 for an example of a real-world use case), and (c) synthetic task/dataset generation in privacy-aware applications without revealing the real data or for augmenting a limited number of previous tasks to improve generalization performance.

To tackle the above challenge, this paper presents a novel *implicit process-based meta-learning* (IPML) algorithm (Sec. 3) that, in contrast to existing works, explicitly represents each task as a continuous latent vector and models its probabilistic belief within the highly expressive IP<sup>1</sup> framework (Sec. 2). Unfortunately, meta-training in IPML is computationally challenging due to its need to perform intractable exact IP inference in task adaptation.<sup>2</sup> To resolve this, we propose a novel

<sup>1</sup>An IP (Ma et al., 2019) is a stochastic process such that every finite collection of random variables has an implicitly defined joint prior distribution. Some typical examples of IP include Gaussian processes, Bayesian neural networks, neural processes (Garnelo et al., 2018), among others. An IP is formally defined in Def. 1.

<sup>2</sup>The work of Ma et al. (2019) uses the well-studied Gaussian process as the variational family to perform variational inference in general applications of IP, which sacrifices the flexibility and expressivity of IP by constraining the distributions of the function outputs to be Gaussian. Such a straightforward application of IP to meta-learning has not yielded satisfactory results in our experiments (see Appendix A.4).

*expectation-maximization* (EM) algorithm to perform meta-training (Sec. 3.1): In the E step, we perform task adaptation using the stochastic gradient Hamiltonian Monte Carlo sampling method (Chen et al., 2014) to draw samples from IP posterior beliefs for all meta-training tasks, which eliminates the need to learn a latent encoder (Garnelo et al., 2018). In the M step, we optimize the meta-learning objective w.r.t. the meta-parameters using these samples. Our delicate design of the neural network architecture for meta-training in IPML allows competitive meta-learning performance to be achieved (Sec. 3.2). Our IPML algorithm offers the benefits of being amenable to (a) the characterization of a principled distance measure between tasks using maximum mean discrepancy (Gretton et al., 2012), (b) active task selection without needing the assumption of known task contexts in (Kaddour et al., 2020), and (c) synthetic task generation by modeling task-dependent input distributions (Sec. 3.3).

## 2 BACKGROUND AND NOTATIONS

For simplicity, the inputs (outputs) for all tasks are assumed to belong to the same input (output) space. Consider meta-learning on probabilistic regression tasks:<sup>3</sup> Each task is generated from a task distribution and associated with a dataset  $(\mathcal{X}, \mathbf{y}_{\mathcal{X}})$  where the set  $\mathcal{X}$  and the vector  $\mathbf{y}_{\mathcal{X}} \triangleq (\mathbf{y}_{\mathbf{x}})_{\mathbf{x} \in \mathcal{X}}^{\top}$  denote, respectively, the input vectors and the corresponding noisy outputs

$$\mathbf{y}_{\mathbf{x}} \triangleq f(\mathbf{x}) + \epsilon(\mathbf{x}) \quad (1)$$

which are outputs of an unknown underlying function  $f$  corrupted by an i.i.d. Gaussian noise  $\epsilon(\mathbf{x}) \sim \mathcal{N}(0, \sigma^2)$  with variance  $\sigma^2$ . Let  $f$  be distributed by an *implicit process* (IP), as follows:

**Definition 1** (Implicit process for meta-learning). *Let the collection of random variables  $f(\cdot)$  denote an IP parameterized by meta-parameters  $\theta$ , that is, every finite collection  $\{f(\mathbf{x})\}_{\mathbf{x} \in \mathcal{X}}$  has a joint prior distribution  $p(\mathbf{f}_{\mathcal{X}} \triangleq (f(\mathbf{x}))_{\mathbf{x} \in \mathcal{X}}^{\top})$  implicitly defined by the following generative model:*

$$\mathbf{z} \sim p(\mathbf{z}), \quad f(\mathbf{x}) \triangleq g_{\theta}(\mathbf{x}, \mathbf{z}) \quad (2)$$

for all  $\mathbf{x} \in \mathcal{X}$  where  $\mathbf{z}$  is a latent task vector to be explained below and generator  $g_{\theta}$  can be an arbitrary model (e.g., deep neural network) parameterized by meta-parameters  $\theta$ .

Definition 1 defines valid stochastic processes if  $\mathbf{z}$  is finite dimensional (Ma et al., 2019). Though, in reality, a task may follow an unknown distribution, we assume the existence of an unknown function that maps each task to a latent task vector  $\mathbf{z}$  satisfying the desired known distribution  $p(\mathbf{z})$ , like in (Kaddour et al., 2020).<sup>4</sup> Using  $p(\mathbf{y}_{\mathcal{X}}|\mathbf{f}_{\mathcal{X}}) = \mathcal{N}(\mathbf{f}_{\mathcal{X}}, \sigma^2 \mathbf{I})$  (1) and the IP prior belief  $p(\mathbf{f}_{\mathcal{X}})$  from Def. 1, we can derive the marginal likelihood  $p(\mathbf{y}_{\mathcal{X}})$  by marginalizing out  $\mathbf{f}_{\mathcal{X}}$ .

**Remark 1.** Two sources of uncertainty exist in  $p(\mathbf{y}_{\mathcal{X}})$ : *Aleatoric uncertainty* in  $p(\mathbf{y}_{\mathcal{X}}|\mathbf{f}_{\mathcal{X}})$  reflects the noise (i.e., modeled in (1)) inherent in the dataset, while *epistemic uncertainty* in the IP prior belief  $p(\mathbf{f}_{\mathcal{X}})$  reflects the model uncertainty arising from the *latent task prior belief*  $p(\mathbf{z})$  in (2).<sup>5</sup>

Let the sets  $\mathcal{T}$  and  $\mathcal{T}_*$  denote the meta-training and meta-testing tasks, respectively. Following the convention in (Finn et al., 2018; Gordon et al., 2019; Ravi & Beatson, 2018; Yoon et al., 2018), for each meta-training task  $t \in \mathcal{T}$ , we consider a support-query (or train-test) split of its dataset  $(\mathcal{X}_t, \mathbf{y}_{\mathcal{X}_t})$  into the *support set* (or training dataset)  $(\mathcal{X}_t^s, \mathbf{y}_{\mathcal{X}_t^s})$  and *query set* (or test/evaluation dataset)  $(\mathcal{X}_t^q, \mathbf{y}_{\mathcal{X}_t^q})$  where  $\mathcal{X}_t = \mathcal{X}_t^s \cup \mathcal{X}_t^q$  and  $\mathcal{X}_t^s \cap \mathcal{X}_t^q = \emptyset$ . Specifically, for a  $N$ -way  $K$ -shot classification problem, the support set has  $K$  examples per class and  $N$  classes in total.

Meta-learning can be defined as an optimization problem (Finn et al., 2017; 2018) and its goal is to learn meta-parameters  $\theta$  that maximize the following objective defined over all meta-training tasks:

$$\mathcal{J}_{\text{meta}} \triangleq \log \prod_{t \in \mathcal{T}} p(\mathbf{y}_{\mathcal{X}_t^q} | \mathbf{y}_{\mathcal{X}_t^s}) = \sum_{t \in \mathcal{T}} \log \int p(\mathbf{y}_{\mathcal{X}_t^q} | \mathbf{f}_{\mathcal{X}_t^q}) p(\mathbf{f}_{\mathcal{X}_t^q} | \mathbf{y}_{\mathcal{X}_t^s}) d\mathbf{f}_{\mathcal{X}_t^q}. \quad (3)$$

Task *adaptation*  $p(\mathbf{f}_{\mathcal{X}_t^q} | \mathbf{y}_{\mathcal{X}_t^s})$  is performed via IP inference after observing the support set:

$$p(\mathbf{f}_{\mathcal{X}_t^q} | \mathbf{y}_{\mathcal{X}_t^s}) = \int_{\mathbf{z}} p(\mathbf{f}_{\mathcal{X}_t^q} | \mathbf{z}) p(\mathbf{z} | \mathbf{y}_{\mathcal{X}_t^s}) d\mathbf{z}. \quad (4)$$

<sup>3</sup>We defer the discussion of meta-learning on probabilistic classification tasks using the robust-max likelihood (Hernández-Lobato et al., 2011) to Appendix A.1.

<sup>4</sup> $p(\mathbf{z})$  is often assumed to be a simple distribution like multivariate Gaussian  $\mathcal{N}(\mathbf{0}, \mathbf{I})$  (Garnelo et al., 2018).

<sup>5</sup>Our work here considers a point estimate of meta-parameters  $\theta$  instead of a Bayesian treatment of  $\theta$  (Finn et al., 2018; Yoon et al., 2018). This allows us to interpret the epistemic uncertainty in  $p(\mathbf{f}_{\mathcal{X}})$  via  $p(\mathbf{z})$  directly.

The objective  $\mathcal{J}_{\text{meta}}$  (3) is the “test” likelihood on the query set, which reflects the idea of “learning to learn” by assessing the effectiveness of “learning on the support set” through the query set. An alternative interpretation views  $p(\mathbf{f}_{\mathcal{X}_t^q} | \mathbf{y}_{\mathcal{X}_t^s})$  as an “informative prior” after observing the support set. The objective  $\mathcal{J}_{\text{meta}}$  (3) is also known as the Bayesian held-out likelihood (Gordon et al., 2019). In a meta-testing task, adaptation is also performed via IP inference after observing its support set and evaluated on its query set. Similar to GP or any stochastic process, the input vectors of the dataset are assumed to be known/fixed beforehand. We will relax this assumption by allowing them to be unknown when our IPML algorithm is exploited for synthetic task generation (Sec. 3.3).

### 3 IMPLICIT PROCESS-BASED META-LEARNING (IPML)

#### 3.1 EXPECTATION MAXIMIZATION (EM) ALGORITHM FOR IPML

Recall that task adaptation requires evaluating  $p(\mathbf{f}_{\mathcal{X}_t^q} | \mathbf{y}_{\mathcal{X}_t^s})$  (4). From Def. 1, if generator  $g_\theta$  (2) can be an arbitrary model (e.g., deep neural network), then  $p(\mathbf{f}_{\mathcal{X}_t^q} | \mathbf{y}_{\mathcal{X}_t^s})$  and  $p(\mathbf{f}_{\mathcal{X}_t^q})$  cannot be evaluated in closed form and have to be approximated by samples. Inspired by the Monte Carlo EM algorithm (Wei & Tanner, 1990) which utilizes posterior samples to obtain a maximum likelihood estimate of some hyperparameters, we propose an EM algorithm for IPML: The E step uses the *stochastic gradient Hamiltonian Monte Carlo* (SGHMC) sampling method to draw samples from  $p(\mathbf{f}_{\mathcal{X}_t^q} | \mathbf{y}_{\mathcal{X}_t^s})$  (4), while the M step maximizes the meta-learning objective  $\mathcal{J}_{\text{meta}}$  (3) w.r.t. meta-parameters  $\theta$ :

**Expectation (E) step.** Note that since  $\mathbf{f}_{\mathcal{X}_t^q} = (g_\theta(\mathbf{x}, \mathbf{z}))_{\mathbf{x} \in \mathcal{X}_t^q}^\top$  (2), no uncertainty exists in  $p(\mathbf{f}_{\mathcal{X}_t^q} | \mathbf{z})$  in (4). So,  $p(\mathbf{f}_{\mathcal{X}_t^q} | \mathbf{y}_{\mathcal{X}_t^s})$  can be evaluated using the same generator  $g_\theta$  (2) and the *latent task posterior belief*  $p(\mathbf{z} | \mathbf{y}_{\mathcal{X}_t^s})$ , as follows:

**Remark 2.** Drawing samples from  $p(\mathbf{f}_{\mathcal{X}_t^q} | \mathbf{y}_{\mathcal{X}_t^s})$  is thus equivalent to first drawing samples of  $\mathbf{z}$  from  $p(\mathbf{z} | \mathbf{y}_{\mathcal{X}_t^s})$  and then passing them as inputs to generator  $g_\theta$  to obtain samples of  $\mathbf{f}_{\mathcal{X}_t^q}$ . Hence, given a task  $t$ , adaptation  $p(\mathbf{f}_{\mathcal{X}_t^q} | \mathbf{y}_{\mathcal{X}_t^s})$  (4) essentially reduces to a task identification problem by performing IP inference to obtain the latent task posterior belief  $p(\mathbf{z} | \mathbf{y}_{\mathcal{X}_t^s})$ . This is a direct consequence of epistemic uncertainty arising from  $p(\mathbf{z} | \mathbf{y}_{\mathcal{X}_t^s})$  and  $p(\mathbf{z})$  (Remark 1).

In general,  $p(\mathbf{z} | \mathbf{y}_{\mathcal{X}_t^s})$  also cannot be evaluated in closed form. Instead of using *variational inference* (VI) and approximating  $p(\mathbf{z} | \mathbf{y}_{\mathcal{X}_t^s})$  with a potentially restrictive variational distribution (Garnelo et al., 2018; Kaddour et al., 2020; Ma et al., 2019), we draw samples from  $p(\mathbf{z} | \mathbf{y}_{\mathcal{X}_t^s})$  using SGHMC (Chen et al., 2014). SGHMC introduces an auxiliary random vector  $\mathbf{r}$  and samples from a joint distribution  $p(\mathbf{z}, \mathbf{r} | \mathbf{y}_{\mathcal{X}_t^s})$  following the Hamiltonian dynamics (Brooks et al., 2011; Neal, 1993):  $p(\mathbf{z}, \mathbf{r} | \mathbf{y}_{\mathcal{X}_t^s}) \propto \exp(-U(\mathbf{z}) - 0.5\mathbf{r}^\top \mathbf{M}^{-1} \mathbf{r})$  where the negative log-probability  $U(\mathbf{z}) \triangleq -\log p(\mathbf{z} | \mathbf{y}_{\mathcal{X}_t^s})$  resembles the potential energy and  $\mathbf{r}$  resembles the momentum. SGHMC updates  $\mathbf{z}$  and  $\mathbf{r}$ , as follows:

$$\Delta \mathbf{z} = \alpha \mathbf{M}^{-1} \mathbf{r}, \quad \Delta \mathbf{r} = -\alpha \nabla_{\mathbf{z}} U(\mathbf{z}) - \alpha \mathbf{C} \mathbf{M}^{-1} \mathbf{r} + \mathcal{N}(\mathbf{0}, 2\alpha(\mathbf{C} - \mathbf{B}))$$

where  $\alpha$ ,  $\mathbf{C}$ ,  $\mathbf{M}$ , and  $\mathbf{B}$  are the step size, friction term, mass matrix, and Fisher information matrix, respectively.<sup>6</sup> Note that  $\nabla_{\mathbf{z}} U(\mathbf{z}) = -\nabla_{\mathbf{z}} \log p(\mathbf{z} | \mathbf{y}_{\mathcal{X}_t^s}) = -\nabla_{\mathbf{z}} \log p(\mathbf{z}, \mathbf{y}_{\mathcal{X}_t^s}) = -\nabla_{\mathbf{z}} [\log p(\mathbf{y}_{\mathcal{X}_t^s} | \mathbf{f}_{\mathcal{X}_t^s} = (g_\theta(\mathbf{x}, \mathbf{z}))_{\mathbf{x} \in \mathcal{X}_t^s}^\top) + \log p(\mathbf{z})]$  can be evaluated tractably.

**Maximization (M) step.** We optimize  $\mathcal{J}_{\text{meta}}$  (3) w.r.t.  $\theta$  using samples of  $\mathbf{z}$ . The original objective  $\mathcal{J}_{\text{meta}} = \sum_{t \in \mathcal{T}} \log(\mathbb{E}_{p(\mathbf{z} | \mathbf{y}_{\mathcal{X}_t^s})} [p(\mathbf{y}_{\mathcal{X}_t^q} | \mathbf{f}_{\mathcal{X}_t^q} = (g_\theta(\mathbf{x}, \mathbf{z}))_{\mathbf{x} \in \mathcal{X}_t^q}^\top)])$  is not amenable to stochastic optimization with data minibatches, which is usually not an issue in a few-shot learning setting. When a huge number of data points and samples of  $\mathbf{z}$  are considered, we can resort to optimizing the lower bound  $\mathcal{J}_{\text{s-meta}}$  of  $\mathcal{J}_{\text{meta}}$  by applying the Jensen’s inequality:

$$\mathcal{J}_{\text{meta}} \geq \mathcal{J}_{\text{s-meta}} \triangleq \sum_{t \in \mathcal{T}} \mathbb{E}_{p(\mathbf{f}_{\mathcal{X}_t^q} | \mathbf{y}_{\mathcal{X}_t^s})} [\log p(\mathbf{y}_{\mathcal{X}_t^q} | \mathbf{f}_{\mathcal{X}_t^q})] = \sum_{t \in \mathcal{T}} \mathbb{E}_{p(\mathbf{z} | \mathbf{y}_{\mathcal{X}_t^s})} [\log p(\mathbf{y}_{\mathcal{X}_t^q} | \mathbf{f}_{\mathcal{X}_t^q})].$$

<sup>6</sup>The sampler hyperparameters  $\alpha$ ,  $\mathbf{C}$ ,  $\mathbf{M}$ , and  $\mathbf{B}$  are set according to the auto-tuning method of Springenberg et al. (2016) which has been verified to work well in our experiments; more details are given in Appendix A.2.1.

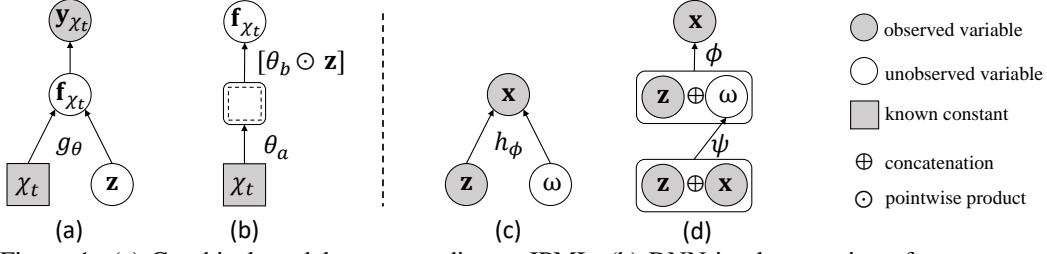


Figure 1: (a) Graphical model corresponding to IPML. (b) DNN implementation of generator  $g_\theta$  where  $\theta \triangleq (\theta_a, \theta_b)$  and  $\theta_a$  can be convolutions to obtain high-level representations of the input vector, while  $\theta_b$  is the last DNN layer’s parameters which are masked by  $z$  during the forward passes. (c) Graphical model corresponding to input generation by X-Net. (d) CVAE implementation of X-Net (i.e., decoder neural network with parameters  $\phi$ ).

### 3.2 ARCHITECTURE DESIGN FOR META-TRAINING

Our generator  $g_\theta$  is implemented using a *deep neural network* (DNN) parameterized by meta-parameters  $\theta$ . Under this setup, we have empirically observed that the design of the coupling of  $z$  with the DNN  $g_\theta(\mathbf{x}, \cdot)$  is crucial to achieving competitive performance of our IPML algorithm. A naive design by concatenating  $z$  with  $\mathbf{x}$  (or higher-level abstractions of  $\mathbf{x}$ ) as a contextual input during forward passes has not worked well as the resulting gradients w.r.t.  $z$  may not have provided enough guidance for SGHMC to learn a sufficiently useful representation of  $z$  in meta-training.

To this end, inspired by the *attention* mechanism (Vaswani et al., 2017) and *dropout* method (Srivastava et al., 2014), we introduce a design of the coupling by applying  $z$  as a *mask* to the last DNN layer’s parameters: The last DNN layer’s parameters are first masked by  $z$  (i.e., point-wise product with  $z$ ), as illustrated in Figs. 1a and 1b. Different tasks can now be distinguished by different masks, hence resembling different attentions on the last DNN layer’s connections during forward propagation. We adopt soft masks<sup>7</sup> (i.e., continuous values) instead of hard masks (i.e., either 0 or 1). Such a design of the coupling is empirically demonstrated to be effective in our experiments (Appendix A.4.3).

### 3.3 ARCHITECTURE DESIGN FOR SYNTHETIC TASK GENERATION

Recall the assumption of known/fixed input vectors in  $\mathcal{X}_t$  in the last paragraph of Sec. 2,<sup>8</sup> which we will have to relax here. Synthetic task generation can be performed by the following procedure if  $\mathbf{x}$  is task-independent (e.g.,  $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x})p(\mathbf{z})$ ): After meta-training is completed (Sec. 2), draw a sample of latent task vector  $\mathbf{z} \sim p(\mathbf{z})$ , draw samples of  $\mathbf{x} \sim p(\mathbf{x})$  to form  $\mathcal{X}_t$ , and then generate noisy outputs  $\mathbf{y}_{\mathcal{X}_t} = (g_\theta(\mathbf{x}, \mathbf{z}) + \epsilon(\mathbf{x}))_{\mathbf{x} \in \mathcal{X}_t}^\top$  to obtain the dataset  $(\mathcal{X}_t, \mathbf{y}_{\mathcal{X}_t})$  for task  $t$ .

When  $\mathbf{x}$  is task-dependent (e.g., for image classifications of different objects,  $p(\mathbf{x}, \mathbf{z}) \neq p(\mathbf{x})p(\mathbf{z})$ ), not modeling  $p(\mathbf{x}|\mathbf{z})$  limits the ability to generate  $t$ -dependent  $\mathcal{X}_t$ . To resolve this, our IPML algorithm includes an *X-generative network* (X-Net):  $\mathbf{x} \triangleq h_\phi(\mathbf{z}, \omega)$  that learns to generate an input vector  $\mathbf{x}$  given samples of the latent task vector  $\mathbf{z}$  and random vector  $\omega \sim p(\omega) = \mathcal{N}(\mathbf{0}, \mathbf{I})$  where  $\omega$  models the diversity of the input distribution given a fixed task represented by the sample of  $\mathbf{z}$ . There are several options to implement X-Net: Note that during the training of X-Net, both  $\mathcal{X}_t$  and the samples of  $\mathbf{z} \sim p(\mathbf{z}|\mathbf{y}_{\mathcal{X}_t^s})$  for all meta-training task  $t \in \mathcal{T}$  are available. So, generative models such as the *conditional variational autoencoder* (CVAE) (Sohn et al., 2015) or conditional generative adversarial networks (Mirza & Osindero, 2014) are suitable for X-Net as they can utilize  $\mathbf{z}$  as the *contextual* information. Our work here uses (the decoder of) CVAE to implement X-Net. Figs. 1c and 1d illustrate such a design. We have empirically observed that a simple concatenation with  $\mathbf{z}$  suffices here as our delicate architecture design for meta-training (Sec. 3.2) can yield a useful representation of  $\mathbf{z}$  for training X-Net well. Further details and a method to ensure balanced data generation are given in Appendix A.5. The training objective for synthetic task generation is the

<sup>7</sup>The latent task prior belief  $p(\mathbf{z})$  is thus assumed to be a multivariate Gaussian  $\mathcal{N}(\mathbf{1}, \mathbf{I})$ .

<sup>8</sup>This assumption is reasonable for meta-training since only  $p(\mathbf{y}_{\mathcal{X}})$  (and not  $p(\mathbf{x})$ ) needs to be modeled.

empirical lower bound (Sohn et al., 2015) of VI on  $p(\omega|\mathbf{x}, \mathbf{z})$ :

$$\mathcal{J}_X \triangleq \sum_{t \in \mathcal{T}} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}|\mathbf{y}_{\mathcal{X}_t^s})} \left[ |\mathcal{X}_t|^{-1} \sum_{\mathbf{x} \in \mathcal{X}_t} (\mathbb{E}_{q_\psi(\omega|\mathbf{x}, \mathbf{z})} [\log p_\phi(\mathbf{x}|\mathbf{z}, \omega)] - D_{\text{KL}}[q_\psi(\omega|\mathbf{x}, \mathbf{z}) \| p(\omega)]) \right]$$

where  $\phi$  and  $\psi$  are, respectively, the parameters of X-Net (decoder neural network) and the encoder neural network, and  $D_{\text{KL}}$  denotes the KL distance. In the training of X-Net, we sample one  $\mathbf{z}$  per update. We also sample one  $\omega$  per update to train with reparameterization tricks. Algorithms 1 and 2 describe meta-training (with training of X-Net) and synthetic task generation, respectively.

---

**Algorithm 1: IPML: Meta-Training**


---

```

while not converged do
  Sample task  $t$  from  $\mathcal{T}$ 
  E step: Sample  $\{\mathbf{z}_1, \dots, \mathbf{z}_n\}$  with SGHMC
  M step: Sample  $\mathbf{z}$  from  $\{\mathbf{z}_1, \dots, \mathbf{z}_n\}$ 
            $\theta \leftarrow \theta + \eta \nabla_\theta \mathcal{J}_{\text{meta}}$ 
  Update X-Net with  $\mathbf{z}$  and  $\mathcal{X}_t$ :
            $\phi \leftarrow \phi + \eta \nabla_\phi \mathcal{J}_X, \quad \psi \leftarrow \psi + \eta \nabla_\psi \mathcal{J}_X$ 
return  $\theta, \phi, \psi$ 

```

---



---

**Algorithm 2: Synthetic Task Generation**


---

```

Sample  $\mathbf{z} \sim p(\mathbf{z})$ 
Initialize synthetic task  $t$  and  $\mathcal{X}_t = \emptyset$ 
for  $i = 1, \dots$ , final size of  $\mathcal{X}_t$  do
  Sample  $\omega \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
  Compute  $\mathbf{x} = h_\phi(\mathbf{z}, \omega)$ 
  Compute  $y_{\mathbf{x}} = g_\theta(\mathbf{x}, \mathbf{z}) + \epsilon(\mathbf{x})$ 
   $(\mathcal{X}_t, \mathbf{y}_{\mathcal{X}_t}) \leftarrow (\mathcal{X}_t \cup \{\mathbf{x}\}, \mathbf{y}_{\mathcal{X}_t \cup \{\mathbf{x}\}})$ 
return  $(\mathcal{X}_t, \mathbf{y}_{\mathcal{X}_t})$  for task  $t$ 

```

---

## 4 EXPERIMENTS AND DISCUSSION

**Benchmark datasets: sinusoid regression and few-shot image classification.** We first empirically evaluate the performance of our IPML algorithm against that of several Bayesian meta-learning baselines like the *neural process* (NP) (Garnelo et al., 2018), *Bayesian model-agnostic meta-learning* (BMAML) (Yoon et al., 2018), PLATIPUS (Finn et al., 2018), and *amortized Bayesian meta-learning* (ABML) (Ravi & Beatson, 2018) on benchmark meta-learning datasets. For few-shot image classification, we also empirically compare IPML with a strong baseline: *prototypical network* (PN) (Snell et al., 2017). We run experiments on three datasets: sinusoid, Omniglot (Lake et al., 2011), and mini-ImageNet (Ravi & Larochelle, 2017). Sinusoid is a regression task of sine waves with uniformly sampled amplitude in  $[0.1, 5.0]$ , phase in  $[0, \pi]$ , and input  $\mathbf{x}$  in  $[-5, 5]$ . The generator of IPML and the baseline regressors are neural networks with 2 hidden layers of size 40 with ReLU nonlinearities. The Omniglot dataset consists of 20 instances of 1623 characters from 50 different alphabets. The mini-ImageNet dataset involves 64 training classes, 12 validation classes, and 24 test classes. For Omniglot and mini-ImageNet, our implementation and baselines all use the same data pre-processing, same train-test split, and same data augmentation as that in (Finn et al., 2017). The generator of IPML and the baseline classifiers are convolutional neural networks with 4 modules of  $3 \times 3$  convolutions and 64 filters, followed by batch normalization, ReLU nonlinearities, and strided convolutions (Omniglot) or  $2 \times 2$  max-pooling (mini-ImageNet). More details of the experimental settings can be found in Appendix A.2.2.

For sinusoid regression (Table 1), IPML outperforms MAML and BMAML by a fair margin. For Omniglot (Table 2), IPML is competitive with MAML and PN. For mini-ImageNet (Table 3), IPML outperforms MAML and all tested Bayesian meta-learning algorithms,<sup>9</sup> while being competitive with PN. PN achieves a higher classification accuracy for 1-shot 20-way Omniglot and 5-shot 5-way mini-ImageNet because PN utilizes more information from the extra classes during training (Snell et al., 2017). Specifically, though meta-testing involves  $N$ -way classification for all tested algorithms, the training of PN requires more than  $N$  classes, that is, 60-way classification which is also the setting adopted in (Snell et al., 2017). As a result, since PN utilizes more information from the extra classes during training, it is reasonable to expect that PN achieves a higher classification accuracy at times. Overall, IPML is effective for benchmark datasets.

For both sinusoid regression (Table 1) and Omniglot (Table 2), NP performs unsatisfactorily as compared to IPML, likely because (a) it performs amortized variational inference of  $\mathbf{z}$  through a heavily parameterized encoder which may introduce optimization difficulties and overfitting during meta-training, and (b) the encoder of NP takes in the simple concatenation of  $(\mathbf{x}, y_{\mathbf{x}})$  and thus does not explicitly capture the  $\mathbf{x} \rightarrow y_{\mathbf{x}}$  relationship in the support set.<sup>10</sup>

<sup>9</sup>Some of the results are taken from (Finn et al., 2018; Nguyen et al., 2020; Yoon et al., 2018). The 5-shot 5-way results for PLATIPUS and ABML are missing because there are no publicly available implementations.

<sup>10</sup>An ablation study of the limitations of NP can be found in Appendix A.8.

Table 1: *Mean square error* (MSE) on few-shot sinusoid regression.

	Sinusoid 5-shot	Sinusoid 10-shot
NP	0.460	0.264
MAML	0.712	0.287
BMAML	0.409	0.200
IPML(Ours)	<b>0.373</b>	<b>0.123</b>

Table 2: Few-shot classification accuracy (%) on held-out Omniglot characters.

	Omniglot 1-shot 5-way	Omniglot 1-shot 20-way
NP	95.9	55.3
MAML	98.7	92.5
PN	98.8	<b>96.0</b>
IPML(Ours)	<b>98.8</b>	94.0

Table 3: Few-shot classification accuracy (%) on mini-ImageNet test set.

	mini-ImageNet 1-shot 5-way	mini-ImageNet 5-shot 5-way
MAML	48.6	65.9
PN	49.4	<b>68.2</b>
PLATIPUS	50.1	-
BMAML	49.1	64.2
ABML	45.0	-
IPML(Ours)	<b>50.5</b>	67.6

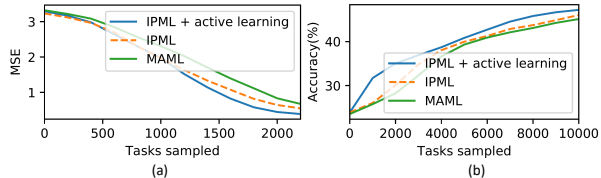


Figure 2: Results of active task selection on (a) 5-shot sinusoid and (b) 1-shot 5-way mini-ImageNet.

**Active task selection.** We can evaluate the effectiveness of the uncertainty measure arising from latent task posterior belief  $p(\mathbf{z}|\mathbf{y}_{\mathcal{X}_t^s})$  by performing active task selection. Unlike previous works (Yoon et al., 2018; Finn et al., 2018) that can only perform active learning by querying data points, IPML can perform active learning by querying *tasks* and does not need the assumption of known task contexts in (Kaddour et al., 2020). In every iteration, a set of tasks are proposed with only the support set  $(\mathcal{X}_t^s, \mathbf{y}_{\mathcal{X}_t^s})$  given; in image classification, it is usually one-shot. IPML will select among them the task with the maximum variance in  $p(\mathbf{z}|\mathbf{y}_{\mathcal{X}_t^s})$  (with samples from the E step/SGHMC):  $\arg \max_t \text{Var}(\mathbf{z}|\mathbf{y}_{\mathcal{X}_t^s})$ , and request for its query set to perform meta-training. This corresponds to a variance-based active task selection criterion. We test on both sinusoid regression and mini-ImageNet classification. Fig. 2 shows that the performance of IPML with active task selection improves over that of both MAML or IPML without active task selection, that is, it reaches a given MSE/accuracy with less training tasks. This shows that the uncertainty measure arising from  $p(\mathbf{z}|\mathbf{y}_{\mathcal{X}_t^s})$  can be exploited to benefit meta-training.

**Measuring distance between tasks using latent task representation.** A most interesting question yet to be answered is the following: Does IPML learn a useful latent task representation? IPML learns to model the task through  $\mathbf{z}$ . If IPML learns the correct representation, then it can reflect patterns of task distribution in the latent space. While a solid criterion for assessing the correctness of learned latent task representation is hard to define, we can resort to an oracle (e.g., human expert with prior knowledge in designing the tasks). Our visualization of the latent task representation and quantitative evaluation of distance measure between tasks using *maximum mean discrepancy* (MMD) (Gretton et al., 2012) provide ways to assess the correctness of the learned task representation. We denote the set of samples from  $p(\mathbf{z}|\mathbf{y}_{\mathcal{X}_t^s})$  as  $\mathcal{Z}_t$ . The MMD between tasks  $t_1$  and  $t_2$  can be calculated using

$$\text{MMD}[\mathcal{H}, t_1, t_2] \triangleq \sup_{\kappa \in \mathcal{H}} \left( |\mathcal{Z}_{t_1}|^{-1} \sum_{\mathbf{z} \in \mathcal{Z}_{t_1}} \kappa(\mathbf{z}) - |\mathcal{Z}_{t_2}|^{-1} \sum_{\mathbf{z} \in \mathcal{Z}_{t_2}} \kappa(\mathbf{z}) \right)$$

where  $\mathcal{H}$  is a unit ball in the reproducing kernel Hilbert space with a radial basis function kernel.

We conduct experiments with the following 5-way 1-shot settings. **Setting A:** For subsampled Omniglot, we applied one rotation out of 4 possibilities ( $0, \pi/2, \pi, 3\pi/2$ ) uniformly across all the input images for each sampled task.<sup>11</sup> **Setting B:** For subsampled mini-ImageNet, a random artistic filter (normal, brighten, or darken) is applied for each sampled task. **Setting C:** For subsampled mini-ImageNet, a random artistic filter (3 different types of hue) is applied for each sampled task. **Setting D:** For subsampled mini-ImageNet, a random zooming (no zooming, zooming 3 times, or zooming 10 times) is applied for each sampled task. **Setting E:** On subsampled mini-ImageNet, a random artistic filter (normal, low contrast, or high contrast) is applied for each sampled task. **Setting A** has 4 types of tasks while **settings B to E** result in 3 types of tasks.

<sup>11</sup>In the previous experiment, the Omniglot dataset is augmented with rotations, but is random across the classes in a single task.

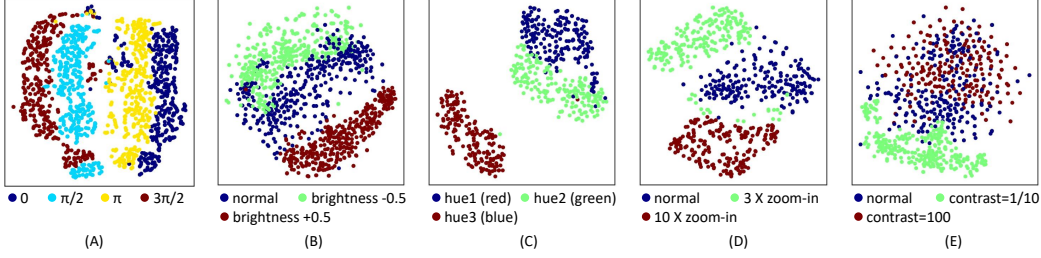


Figure 3: Visualization of latent task embeddings from settings A to E.

Table 4: Values of MMD metric between 4 types of tasks for Omniglot (setting A). Larger value means larger dissimilarity.

Rotations	0	$\pi/2$	$\pi$	$3\pi/2$
0	0	1.166	0.594	1.134
$\pi/2$	1.166	0	0.913	0.596
$\pi$	0.594	0.913	0	0.917
$3\pi/2$	1.134	0.596	0.917	0

Table 5: Results of meta-testing for training with real and generated tasks.

Train on	Accuracy (%)
real	73.83
generated	78.33
real + generated	88.16

For each setting mentioned above, we first train our models in IPML to converge, and then sample tasks from their latent task posterior beliefs (i.e., one sample of  $\mathbf{z}$  per task). Finally, we visualize their latent task embeddings in the 2D space using TSNE (van der Maaten & Hinton, 2008). Furthermore, for setting A, we evaluate the distance measure between tasks using the well-known MMD metric with radial basis function kernels on the  $\mathbf{z}$  samples. It can be observed from Fig. 3 and Table 4 that IPML successfully distinguishes 4 types of rotations for Omniglot. Both Fig. 3 and Table 4 contemporaneously show that flipping upside down (i.e., either right half of the embedding  $0 \rightleftharpoons \pi$  or left half of the embedding  $\pi/2 \rightleftharpoons 3\pi/2$ ) are reckoned to be closer tasks compared with rotation of  $\pi/2$ , thus revealing that our visualization and evaluation of distance measure between tasks are in accordance. From Fig. 3B to Fig. 3D, IPML successfully distinguishes different types of transformations on the tasks while revealing interesting facts: for example, tasks of high brightness are more isolated from that of low or normal brightness. Fig. 3E shows that tasks of low contrast are more distinct from that of normal or high contrast. The values of MMD metric for settings B to E and more details of the experiments are provided in Appendix A.6. On the overall, both the visualization and evaluation of distance measure between tasks reveal that IPML successfully learns useful latent task representations and even provides interesting insights.

**Synthetic task generation for Omniglot.** We assess the usefulness of latent task representation  $\mathbf{z}$  by performing synthetic task generation. The training tasks we consider are three types of sub-sampled binary classifications: classification of characters A vs. B, B vs. C, and C vs. A, as in Fig. 4a. During meta-learning, we train a X-Net concurrently to learn to generate task-related input images (Sec. 3.3). The CVAE implementation of X-Net contains a decoder neural network with 3

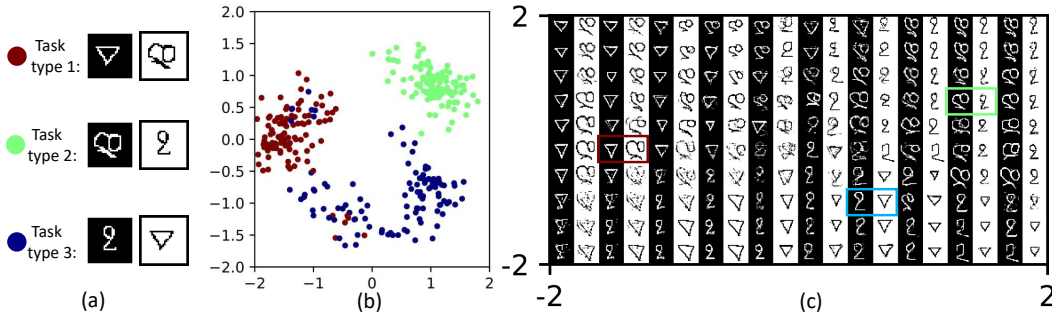


Figure 4: (a) TSNE visualization of (samples of) 3 types of binary classification tasks; images of black/white background are black/white samples ( $y_{\mathbf{x}} = 1/y_{\mathbf{x}} = 0$ ). (b) Visualization of latent embedding of real tasks in (normalized)  $\mathbf{z}$  space  $[-2, 2]^2$ . (c) Sampled generated task data by walking through the (normalized)  $\mathbf{z}$  space  $[-2, 2]^2$ ; note that the inversion of color is only for visualization to distinguish black and white samples. In training, NO images are inverted.



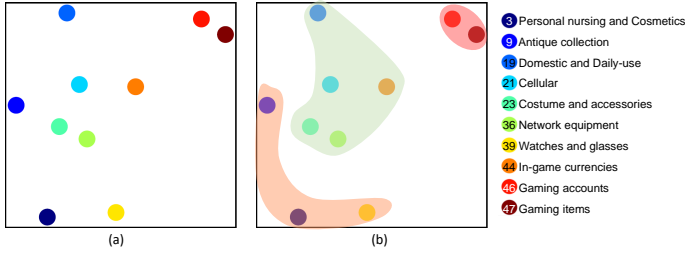


Figure 5: (a) TSNE visualization of latent task embedding of 10 meta-testing categories and (b) their analysis (see main text). Legend shows IDs and names of categories.

Table 6: Averaged meta-testing performance on 10 meta-testing categories.

	Accuracy (%)	F1
IPML	<b>84.5</b>	<b>70.5</b>
Multi-task	84.1	60.5

Table 7: Averaged meta-testing performance on 5 desired categories (IDs 19, 21, 23, 36, 44).

	Accuracy (%)	F1
Setting A	<b>87.4</b>	<b>75.8</b>
Setting B	86.4	74.4

hidden layers of size  $[128, 128, 256]$  and ReLU nonlinearity, and a symmetric design of the encoder. After meta-training is completed, we continue to train the X-Net to converge. In this experiment, the dimension of  $\mathbf{z}$  is set as 2, which further allows walking through such a latent space/embedding to visualize how the generated tasks map to their latent representations. Fig. 4b shows the latent embedding of real tasks. Fig. 4c shows the sampled synthetic tasks by walking through the latent space. It can be observed that X-Net successfully captures the task-dependent input distributions and can generate high-quality data of task type 1, 2, and 3 when sampled from their corresponding latent clusters (see samples of task type 1, 2, and 3 in the colored bounding boxes in Fig. 4c).

We further evaluate the quality of generated tasks by training on it. We hold out half of the images for each character during meta-training to construct the meta-testing tasks. The results are presented in Table 5. When training on both real and generated tasks, we first train on the generated tasks to converge and then train on the real tasks for another 30 iterations. It can be observed that compared to only using real tasks, a higher accuracy is achieved with training merely using generated tasks. When training on both real and generated tasks, a huge boost in accuracy is observed. We conjecture that due to their diversity, generated tasks (i.e., sometimes containing more ambiguous tasks) alleviate overfitting and provide a promising direction on meta-task augmentation.

**Real-world risk detection.** We perform experiments on a real-world risk detection dataset provided by an anonymous e-commerce company. The task is to classify whether an item in the online shop has risks (e.g., fraud, pornography, contraband). Such risks appear in different forms and in different categories (of items). It is hard to detect risks in different categories by training models separately for each category because some categories have only very limited amounts of black samples (i.e.,  $< 50$ ). The similarities of the detected risks in different categories, if discovered, can help improve the performance. Meta-learning is thus a suitable algorithm for its ability to perform (a) detection of risks across different categories of items and (b) adaptation to new categories. The input  $\mathbf{x}$  of the dataset is the text (title and descriptions) embedding obtained from self-supervised learning, while its label is a binary variable indicating whether it contains risks (i.e.,  $y_{\mathbf{x}} = 1$  for black samples and  $y_{\mathbf{x}} = 0$  for white samples). The data are separated by categories of items to yield 47 categories in total. Initially, we hold out 10 categories for meta-testing<sup>12</sup> while the rest are used for meta-training.

Table 6 shows results comparing the performance of IPML vs. a multi-task learning baseline.<sup>13</sup> It can be observed that IPML outperforms multi-task learning, which indicates its stronger ability to generalize to unseen categories. Fig. 5 visualizes the latent task embedding of the 10 meta-testing categories for analysis. IPML learns useful latent task representations: For example, from Fig. 5a, gaming-related categories with IDs 46 and 47 are mapped closely in the latent task space/embedding.

The individual meta-testing performance on the 10 meta-testing categories, which are given in Appendix A.3, can be further examined: For the five categories with IDs 19, 21, 23, 36, and 44 covered by the shaded light green zone in Fig. 5b, IPML outperforms multi-task learning by a large margin. They are mapped to the center of the latent task space (Fig. 5b), which may imply that IPML’s adaptations to them can largely build on previous experiences of the meta-training categories and IPML’s exploitation of such similarities allows their performance to improve over multi-task learning. For

<sup>12</sup>Their category names and IDs are given in Fig. 5.

<sup>13</sup>When testing on an unseen category, multi-task learning performs adaptation by randomly initializing its untied parameters for retraining on the few-shot support data.



the three categories with IDs 3, 9, and 39 covered by the shaded light orange zone, IPML does not have a performance advantage over multi-task learning. For the two categories with IDs 46 and 47 covered by the shaded light pink zone, both IPML and multi-task learning perform unsatisfactorily. As a matter of fact, for IPML, the categories with unsatisfactory performance (i.e., either covered by the shaded light orange or pink zone) are all mapped to be some distance away from the center, which indicates that they are likely considered by IPML as “outlier”/dissimilar tasks.

We further compare meta-learning on (A) the same setting as before by holding out the 10 meta-testing categories vs. (B) training on all categories in setting A as well as the dissimilar ones with IDs 3, 9, 39, 46, and 47. Table 7 shows results on the desired categories with IDs 19, 21, 23, 36, and 44. It can be observed that when a meta-learning model is trained to perform well (during meta-testing) on the desired categories/tasks, training alongside with dissimilar ones can compromise its performance. More details of the experimental settings and data preparation, experimental results, and analysis are given in Appendix A.3. We have also empirically compared the time efficiency of IPML against that of several meta-learning baselines and reported the results in Appendix A.7.

## 5 RELATED WORK

A number of meta-learning algorithms (Finn et al., 2018; Ravi & Beatson, 2018; Yoon et al., 2018) have proposed a Bayesian extension of the MAML framework (Finn et al., 2017). Their difference with IPML is that they model the uncertainty in the predictions with a set of particles (Yoon et al., 2018) or a variational distribution (Finn et al., 2018; Ravi & Beatson, 2018), which does not allow latent task modeling. The work of Rusu et al. (2019) introduces a generative model that decodes latent vectors into the meta-parameters, but does not scale well in the dimension of meta-parameters. In comparison, IPML explicitly represents each task as a latent continuous vector and models its probabilistic belief and is hence scalable in the dimension of meta-parameters. Moreover, MAML-based algorithms usually require evaluating computationally-intensive second-order derivatives of the meta-parameters during meta-training because they approximate the Bayesian inference through an inner loop of gradient descent. Although this issue can be addressed by methods such as first-order approximations (e.g., first-order MAML (Finn et al., 2017), Reptile (Nichol et al., 2018)) or implicit MAML (Rajeswaran et al., 2019) using implicit gradient, these works are not Bayesian. In contrast, our IPML algorithm naturally utilizes Bayes’ rule to perform sampling during Bayesian inference and does not need second-order derivatives.

The work of Kaddour et al. (2020) uses latent information to perform active task selection, but assumes known task-descriptor (task context) which is usually unknown. The work of Garnelo et al. (2018) introduces the first use of stochastic processes (i.e., neural processes) in meta-learning and learns a heavily parameterized encoder to encode a dataset into its latent representation, which might introduce optimization difficulties and overfitting and can only output Gaussian posterior beliefs. In comparison, our IPML algorithm is the first to consider SGHMC in task adaptation/inference of meta-learning, which can capture a non-Gaussian posterior belief to achieve a better performance (Appendix A.4). Our IPML algorithm is also the first to explicitly model task-dependent input distributions, which is lacking in the literature. Such a modeling enables synthetic task generation of complex image classification tasks for the first time.

## 6 CONCLUSION

This paper describes a novel IPML algorithm that, in contrast to existing works, explicitly represents each task as a continuous latent vector and models its probabilistic belief within the highly expressive IP framework. Unlike existing works, IPML offers the benefits of being amenable to (a) the characterization of a principled distance measure between tasks using MMD, (b) active task selection without needing the assumption of known task contexts in (Kaddour et al., 2020), and (c) synthetic task generation of complicated image classifications via modeling of task-dependent input distributions using our X-Net. Empirical evaluation on benchmark datasets shows that IPML outperforms existing Bayesian meta-learning algorithms. We have also empirically demonstrated on an anonymous e-commerce company’s real-world dataset that IPML outperforms the multi-task learning baseline and identifies “outlier”/dissimilar tasks which can degrade meta-testing performance.

## REFERENCES

- S. Brooks, A. Gelman, G. Jones, and X. Meng. *Handbook of Markov chain Monte Carlo*. CRC Press, 2011.
- T. Chen, E. Fox, and C. Guestrin. Stochastic gradient Hamiltonian monte carlo. In *Proc. ICML*, pp. 1683–1691, 2014.
- C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. ICML*, pp. 1126–1135, 2017.
- C. Finn, K. Xu, and S. Levine. Probabilistic model-agnostic meta-learning. In *Proc. NeurIPS*, pp. 9516–9527, 2018.
- M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. Eslami, and Y. W. Teh. Neural processes. arXiv:1807.01622, 2018.
- J. Gordon, J. Bronskill, M. Bauer, S. Nowozin, and R. E. Turner. Meta-learning probabilistic inference for prediction. In *Proc. ICLR*, 2019.
- A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012.
- D. Hernández-Lobato, J. M. Hernández-Lobato, and P. Dupont. Robust multi-class Gaussian process classification. In *Proc. NeurIPS*, pp. 280–288, 2011.
- G. Jerfel, E. Grant, T. Griffiths, and K. A. Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. In *Proc. NeurIPS*, pp. 9119–9130, 2019.
- J. Kaddour, S. Sæmundsson, and M. P. Deisenroth. Probabilistic active meta-learning. In *Proc. NeurIPS*, 2020.
- B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum. One shot learning of simple visual concepts. In *Proc. CogSci*, 2011.
- C. Ma, Y. Li, and J. M. Hernández-Lobato. Variational implicit processes. In *Proc. ICML*, pp. 4222–4233, 2019.
- M. Mirza and S. Osindero. Conditional generative adversarial nets. arXiv:1411.1784, 2014.
- R. M. Neal. Bayesian learning via stochastic dynamics. In *Proc. NeurIPS*, pp. 475–482, 1993.
- C. Nguyen, T. Do, and G. Carneiro. Uncertainty in model-agnostic meta-learning using variational inference. In *Proc. WACV*, pp. 3090–3100, 2020.
- A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. arXiv:1803.02999, 2018.
- A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine. Meta-learning with implicit gradients. In *Proc. NeurIPS*, pp. 113–124, 2019.
- S. Ravi and A. Beaton. Amortized Bayesian meta-learning. In *Proc. ICLR*, 2018.
- S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *Proc. ICLR*, 2017.
- A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell. Meta-learning with latent embedding optimization. In *Proc. ICLR*, 2019.
- J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Proc. NeurIPS*, pp. 4077–4087, 2017.
- K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In *Proc. NeurIPS*, pp. 3483–3491, 2015.
- J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter. Bayesian optimization with robust Bayesian neural networks. In *Proc. NeurIPS*, pp. 4134–4142, 2016.

- N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56): 1929–1958, 2014.
- L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Proc. NeurIPS*, pp. 5998–6008, 2017.
- G. C. Wei and M. A. Tanner. A Monte Carlo implementation of the EM algorithm and the poor man’s data augmentation algorithms. *Journal of the American Statistical Association*, 85(411): 699–704, 1990.
- J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn. Bayesian model-agnostic meta-learning. In *Proc. NeurIPS*, pp. 7332–7342, 2018.

## A APPENDIX: ADDITIONAL DETAILS, EXPERIMENTAL SETTINGS, RESULTS, AND ANALYSIS

### A.1 CLASSIFICATION WITH ROBUST-MAX LIKELIHOOD

Following the work of Hernández-Lobato et al. (2011), the likelihood for the prediction of a data pair  $(\mathbf{x}, y_{\mathbf{x}})$  in a  $N$ -way classification problem given IP output  $\mathbf{f}$  (where this  $N$ -dimensional output is defined as  $\mathbf{f} \triangleq [f_1, f_2, \dots, f_N]$ ) and a binary variable  $a$  (one per data instance to indicate whether an arg max prediction is satisfied or not) is

$$p(y_{\mathbf{x}}|\mathbf{x}, \mathbf{f}, a) = \prod_{c \neq y_{\mathbf{x}}} \Theta(f_{y_{\mathbf{x}}} - f_c)^{1-a} (1/N)^a$$

where  $\Theta(\cdot)$  is the Heaviside step function and the binary variable  $a$  is defined to follow *a priori* a factorizing multivariate Bernoulli distribution:

$$p(a|\rho) \triangleq \text{Bern}(a|\rho) = \rho^a (1 - \rho)^{1-a}$$

such that  $\rho$  is the fraction of training data pairs expected to be outliers. The prior for  $\rho$  is defined as a conjugate beta distribution:

$$p(\rho) \triangleq \text{Beta}(\rho|a_0, b_0) = \frac{\rho^{a_0-1} (1 - \rho)^{b_0-1}}{B(a_0, b_0)}$$

where  $B(\cdot, \cdot)$  is the beta function, and  $a_0$  and  $b_0$  are free hyperparameters which do not have a big impact on the final model, provided that  $b_0 > a_0$  and that they are not too small (Hernández-Lobato et al., 2011). By default, we set  $a_0 = 1$  and  $b_0 = 9$ .

### A.2 DETAILS OF EXPERIMENTAL SETTINGS

#### A.2.1 SAMPLER HYPERPARAMETERS

In practice, running the SGHMC sampling process has two phases: The first is the burn-in phase used to determine the suitable sampler hyperparameters, while the second is the sampling phase which is run using the fixed sampler hyperparameters. In the burn-in phase, the hyperparameters of the sampler are selected using a heuristic auto-tuning approach following that of Springenberg et al. (2016) with initial value of  $\epsilon = 0.03$ ,  $C = 0.05$ , and  $M = \mathbf{I}$ .

Since consecutive samples are highly correlated, we do not choose consecutive samples to perform the M step to prevent the risk of overfitting to those samples. Instead, we randomly select one sample of  $\mathbf{z}$  per task from the sampling phase to carry out optimization in the M step. In the sampling phase, we acquire a total of 40 samples for sinusoid regression, and 5 samples for Omniglot and mini-Imagenet classification.

#### A.2.2 META-TRAINING SETTINGS

For sinusoid regression, we train for a total of 15000 iterations with an Adam optimizer of meta-learning rate (i.e., learning rate of the outer gradient step) 0.001 and meta-batch size 25. For Omniglot and mini-Imagenet classification, we train for a total of 60000 iterations with an Adam optimizer of meta-learning rate 0.001, and meta-batch size is 16 and 4 for Omniglot and mini-Imagenet, respectively. Gradient clipping of  $\pm 10$  is applied in both the SGHMC step and the outer gradient step.

Although IPML does not require any inner gradient step, we have found that adding such inner optimization further improves performance. Thus, in our implementation, we perform inner gradient optimization of the whole model’s parameter after E step, and the M step will differentiate through those inner gradient steps (only in a first-order manner). Note that this does not violate our analysis in Sec. 3.1. Our analysis on the E step is accurate given the current best estimation of the meta-parameter  $\theta$ . However, after knowing the support set, we are able to optimize our current estimation of  $\theta$ . The inner gradient optimization corresponds to such a step of refining our estimation of  $\theta$  and can be interpreted as an inner M step nested in the outer E step. We perform 5 inner gradient steps by default.

### A.2.3 ACTIVE TASK SELECTION SETTINGS

For sinusoid, we consider 25 meta-tasks per iteration and IPML will choose 1 of them according to our proposed active task selection criterion (i.e., variance in samples of  $\mathbf{z}$ ). For mini-Imagenet, we consider 16 meta-tasks per iteration and IPML will choose 1 of them. IPML will store these selected tasks in a buffer and train on them with meta-batch size of 5 and 4 for sinusoid and mini-Imagenet, respectively. Without active task selection, the model will just train on all tasks received.

**Motivating active task selection with a real-world use case.** In our real-world risk detection experiment (Sec. 4), a considerable fraction of black and white labels (i.e., risk or no risk) are labeled manually due to the rapidly evolving nature of risks or fast emergence of new variants of risks. However, for the anonymous e-commerce company, the labeling budget is limited (e.g., a human can only label a fixed amount of items per day). This will require us to select the best sequence of tasks to be labeled to best improve the model. Such problem is crucial and is essentially an active task selection problem that can be tackled using our proposed IPML.

### A.3 EXPERIMENTS ON AN ANONYMOUS E-COMMERCE COMPANY’S RISK DETECTION DATASET

In an anonymous company’s online e-shop, the items advertised by the sellers may contain risks (e.g., fraud, pornography, contraband). Such risks appear in different forms and in different categories (of items). It is hard to detect risks in different categories by training models separately for each category because some categories have only very limited amounts of black samples (i.e.,  $< 50$ ). The similarities of the detected risks in different categories, if discovered, can help improve the performance and the model can quickly adapt to detect risks on new categories of items with only few-shot data given. Meta-learning is thus a suitable algorithm for its ability to perform (a) detection of risks across different categories of items and (b) adaptation to new categories.

**Data preparation and training settings.** We first perform self-supervised learning on the texts of an item using all the raw data available (including items with risk labels or without risk labels). The texts we use are concatenations of an item’s title and descriptions. This allows us to obtain an embedding for each item, which has a dimension of 32768. The embedding will serve as the input  $\mathbf{x}$  while its label is a binary variable indicating whether it contains risks ( $y_{\mathbf{x}} = 1$  for black samples and  $y_{\mathbf{x}} = 0$  for white samples). The data containing all items are separated by categories of items yielding 47 categories in total. Initially, 10 held-out categories are used for meta-testing (see Table 8) while the rest are used for meta-training. Some of the categories have nearly 200000 items while some only have less than 100 items. The fractions of black samples are typically small (average of 14% across 47 categories). We train with a meta-batch size of 10 (each subsampled from a category) and a batch size of 32 (16 support items and 16 query items). During training, we ensure that a task in a minibatch should contain  $> 2$  black samples; otherwise, we skip that task.

Table 8: Ten held-out categories for meta-testing.

Category ID	Category Name
3	Personal nursing and Cosmetics
9	Antique collection
19	Domestic and Daily-use
21	Cellular
23	Costume and accessories
36	Network equipment
39	Watches and glasses
44	In-game currencies
46	Gaming accounts
47	Gaming items

We use a 2-layer neural network with the first layer having 1024 hidden neurons, and the second layer outputs a binary classification. A leaky ReLU activation of slope 0.1 is used in the hidden layer. We choose a multi-task learning baseline for performance comparison, which explicitly ties the parameters on the first layer of neural networks, and extends a separate second layer for binary

Table 9: Meta-testing performance on 10 held-out meta-testing categories for IPML.

Category ID	Accuracy (%)	F1	Recall	Precision
3	85.3	76.1	88.8	66.6
9	83.0	73.8	88.8	63.1
19	90.2	81.4	88.0	75.8
21	94.5	81.7	86.0	77.8
23	89.0	81.3	88.9	75.0
36	79.4	63.1	66.6	60.0
39	93.6	85.7	100.0	75.0
44	84.0	71.4	80.0	64.5
46	72.1	43.9	40.7	47.8
47	74.0	45.8	40.8	52.3
Average	84.5	70.5	76.8	65.8

Table 10: Meta-testing performance on 10 held-out meta-testing categories for multi-task learning baseline.

Category ID	Accuracy (%)	F1	Recall	Precision
3	90.2	79.1	70.3	90.4
9	87.0	75.4	74.4	76.9
19	82.2	57.1	48.1	70.5
21	87.3	69.7	60.3	83.3
23	78.7	55.9	51.8	60.8
36	80.0	52.3	40.7	73.3
39	93.6	83.3	83.3	83.3
44	79.0	53.3	48.2	60.0
46	78.1	38.8	25.9	77.7
47	77.5	41.0	29.6	66.6
Average	84.1	60.5	54.1	73.9

classification (i.e., one for each category). When testing on an unseen category, multi-task learning performs adaptation by randomly initializing its second layer’s parameters for the new category (while tying the first layer’s parameters) and optimizing them on the few-shot support data. For both methods, we train with a learning rate of 0.01 for a total of 8000 iterations. At 8000 iterations, we have observed that the mean meta-training accuracy of IPML on 37 categories is stabilized at around 96.33%. In meta-testing for a particular category, we sampled 200 items (150 white samples and 50 black samples) and split them into a support set of size 100 and a query set of size 100 in every cross-validation trial. The results are averaged over 2 cross-validation trials.

**More discussion on the latent task representation.** As can be seen from Tables 9 and 10, IPML outperforms multi-task learning, which indicates its stronger ability to generalize to unseen categories. Fig. 5 visualizes the latent task embedding of the 10 meta-testing categories for analysis. IPML learns useful latent task representations: For example, from Fig. 5a, gaming-related categories with IDs 46 and 47 are mapped closely in the latent task space/embedding, which coincides with the fact that they both represent risks on categories involving gaming. Another category with ID 44, which is also gaming-related, is mapped closest to them among the rest of 8 categories. IDs 21, 23, and 36 are also mapped closely in the latent space, but a clear interpretation cannot be obtained as they represent quite different categories (i.e., cellular, costume and accessories, and network equipment). Although the white samples of these 3 categories may be considerably dissimilar, we have found that their black samples bear a striking similarity after digging into their raw text (title and descriptions): There is a huge fraction of overlap between their black samples because certain types of malicious sellers (e.g., on fraud and forging of certificates and diplomas) are more likely to advertise their duplicated risk items (black samples) on these three categories, while they are less likely to advertise on other categories such as gaming (e.g., IDs 46 and 47). This shows that our latent task representation and embedding can aid our understanding and instruct us to identify categories that are influenced more by a certain type of malicious sellers.

Table 11: Meta-testing performance on desired categories with IDs 19, 21, 23, 36, and 44 trained without (setting A) or with (setting B) dissimilar tasks.

Category ID Setting	Accuracy (%)		F1		Recall		Precision	
	A	B	A	B	A	B	A	B
19	<b>90.2</b>	89.0	<b>81.4</b>	79.9	88.0	88.0	<b>75.8</b>	73.3
21	<b>94.5</b>	90.4	<b>81.7</b>	80.7	<b>86.0</b>	84.0	<b>77.8</b>	77.7
23	89.0	<b>90.1</b>	81.3	<b>83.2</b>	88.9	<b>92.5</b>	75.0	<b>75.7</b>
36	<b>79.4</b>	76.5	<b>63.1</b>	58.6	<b>66.6</b>	62.9	<b>60.0</b>	54.8
44	84.0	<b>85.0</b>	<b>71.4</b>	69.3	<b>80.0</b>	68.0	64.5	<b>70.8</b>
Average	<b>87.4</b>	86.4	<b>75.8</b>	74.4	<b>81.9</b>	79.8	<b>70.6</b>	70.5

**More results on learning with or without the “outlier”/dissimilar tasks.** From our analysis in Sec. 4, we conjecture that when a meta-learning model is trained to perform well (during meta-testing) on the desired categories/tasks with IDs 19, 21, 23, 36, and 44 which are shown to be similar to previous meta-training tasks, training alongside with dissimilar ones (i.e., with IDs 3, 9, 39, 46, and 47) can compromise its performance. To verify our conjecture, we compare meta-learning on (A) the same setting as before by holding out the 10 meta-testing categories vs. (B) training on all categories in setting A as well as the dissimilar ones with IDs 3, 9, 39, 46, and 47. Table 11 shows detailed results on the individual meta-testing performance on the 10 meta-testing categories. The experimental results agree with our conjecture: By looking at the F1 score, meta-training that additionally includes dissimilar tasks/categories results in degradation of performance in 4 of the 5 desired categories. Thus, to achieve competitive meta-testing performance, one should consider not to include dissimilar tasks during meta-training.

#### A.4 ABLATION STUDY OF THE EFFECTIVENESS OF IPML COMPONENTS

This subsection empirically evaluates the effectiveness of each component of the IPML algorithm.

##### A.4.1 SGHMC VS. VI AND AMORTIZED VI

Table 12: Comparison between IPML-VI vs. IPML on benchmark meta-learning datasets.

	Sinusoid 10-shot (MSE)	Omniglot 1-shot 5-way (Accuracy %)	Omniglot 1-shot 20-way (Accuracy %)	mini-Imagenet 1-shot 5-way (Accuracy %)	mini-Imagenet 5-shot 5-way (Accuracy %)
IPML-VI	0.160	98.7	94.3	50.2	66.6
IPML	0.123	98.8	94.0	50.5	67.6

To obtain the posterior samples from  $p(\mathbf{z}|\mathbf{y}_{\mathcal{X}_t^s})$ , we use SGHMC. Other methods exist to obtain posterior samples from  $p(\mathbf{z}|\mathbf{y}_{\mathcal{X}_t^s})$ . The first method is *variational inference* (VI) that directly constructs a variational distribution for  $p(\mathbf{z}|\mathbf{y}_{\mathcal{X}_t^s})$ . The resulting approach, which we call *IPML with VI* (IPML-VI), replaces SGHMC with VI in our IPML algorithm while keeping its other components unchanged. Unfortunately, such a variational approximation usually yields a biased latent task posterior belief while SGHMC can ideally recover (samples from) the true latent task posterior belief  $p(\mathbf{z}|\mathbf{y}_{\mathcal{X}_t^s})$ . It can be observed from our visualization of the latent task embeddings in Figs. 3 and 4b that the distribution of a cluster of tasks is highly non-Gaussian. We thus conjecture that even for a single task, its distribution is not likely to be Gaussian. So, IPML-VI is limited in its ability to recover the true latent task posterior belief  $p(\mathbf{z}|\mathbf{y}_{\mathcal{X}_t^s})$ . This is empirically supported by the results in Table 12 showing that our IPML algorithm with SGHMC outperforms IPML-VI in 4 of the 5 test cases.

The other method that can obtain posterior samples from  $p(\mathbf{z}|\mathbf{y}_{\mathcal{X}_t^s})$  is amortized VI which is used by the *neural process* (NP) (Garnelo et al., 2018): It learns an encoder that can take in the support set and output the variational distribution for  $p(\mathbf{z}|\mathbf{y}_{\mathcal{X}_t^s})$ . As analyzed in Sec. 4, NP utilizing amortized VI performs unsatisfactorily as compared to IPML for both sinusoid regression (Table 1) and Omniglot (Table 2), likely because (a) it employs a heavily parameterized encoder which may introduce optimization difficulties and overfitting during meta-training, and (b) the encoder of NP takes in the



simple concatenation of  $(\mathbf{x}, y_{\mathbf{x}})$  and thus does not explicitly capture the  $\mathbf{x} \rightarrow y_{\mathbf{x}}$  relationship in the support set. An ablation study of the limitations of NP is presented in Appendix A.8.

#### A.4.2 EM ALGORITHM VS. VARIATIONAL GAUSSIAN PROCESS FRAMEWORK

Table 13: Few-shot classification accuracy (%) on held-out Omniglot characters.

	Omniglot 1-shot 5-way	Omniglot 5-shot 5-way
VGPML	62.7	80.4
IPML	98.8	99.5

Using SGHMC, our EM algorithm can directly maximize the original meta-learning objective  $\mathcal{J}_{\text{meta}}$  (3). In contrast, the VI framework directly models  $p(\mathbf{f}_{\mathcal{X}_t^q} | \mathbf{y}_{\mathcal{X}_t^s})$  and maximizes the variational/evidence lower bound (ELBO) of  $\mathcal{J}_{\text{meta}}$  (3). Such a framework typically assumes  $f$  to be distributed by a *Gaussian process* (GP) (Ma et al., 2019) instead of an IP (Def. 1) and we call the resulting framework *variational GP-based meta-learning* (VGPML). Its flexibility is then largely constrained by the GP and its chosen kernel. Table 13 shows that VGPML using the widely used radial basis function kernel performs unsatisfactorily as compared to IPML on few-shot classification, hence reflecting the effectiveness of our proposed EM algorithm for IPML over VGPML.

#### A.4.3 COUPLING OF LATENT TASK VECTOR $\mathbf{z}$ WITH DNN GENERATOR $g_{\theta}$

Table 14: *Mean square error* (MSE) on few-shot sinusoid regression.

	Sinusoid 5-shot	Sinusoid 10-shot
Concatenation	0.817	0.525
Soft mask	0.373	0.123

As mentioned in Sec. 3.2, the design of the coupling of  $\mathbf{z}$  with the DNN  $g_{\theta}(\mathbf{x}, \cdot)$  is crucial to achieving competitive performance of our IPML algorithm. Table 14 shows results comparing (a) the naive design of concatenating  $\mathbf{z}$  with  $\mathbf{x}$  as a contextual input during forward passes vs. (b) our delicate design of applying  $\mathbf{z}$  as a continuous-valued (soft) mask to the last DNN layer’s parameters (Sec. 3.2), the latter of which is used by our IPML algorithm to produce the experimental results in Sec. 4. It can be observed from Table 14 that our delicate design using  $\mathbf{z}$  as a soft mask outperforms the naive design concatenating  $\mathbf{z}$  with  $\mathbf{x}$  by a considerable margin.

#### A.5 DOUBLY-CONTEXTUAL X-NET

In Sec. 3.3, we have introduced X-Net that can generate both synthetic regression or classification tasks. For the more specific case of balanced  $N$ -way classification tasks, we have prior knowledge that  $y_{\mathbf{x}}$  is uniformly distributed over  $[1, \dots, N]$  in a sampled task. Thus, when performing synthetic task generation, our X-Net can be additionally conditioned on  $y_{\mathbf{x}}$  uniformly sampled from  $[1, \dots, N]$  when generating  $\mathbf{x}$ . We refer to such a design as *doubly-contextual X-generative network* (DC X-Net) since it now takes in both  $\mathbf{z}$  and  $y_{\mathbf{x}}$  as contexts. Note that with DC X-Net, we no longer need the IP to generate labels as the labels are sampled in the first place. Although there is now no explicit forward pass in IP during synthetic task generation, this does not make the DC X-Net an independent model of IP. On the contrary, since  $\mathbf{z}$  (i.e., samples obtained from IP) is taken in as a context during training, the learning of DC X-Net can be interpreted as a knowledge distillation process of the IP.

Now, let the DC X-Net:  $\mathbf{x} \triangleq h_{\phi}(y_{\mathbf{x}}, \mathbf{z}, \omega)$  learn to generate an input vector  $\mathbf{x}$  given a sample of the latent task vector  $\mathbf{z}$ , label  $y_{\mathbf{x}}$  (i.e., converted into one-hot encoding), and a sample of the random vector  $\omega \sim p(\omega) = \mathcal{N}(\mathbf{0}, \mathbf{I})$  where  $\omega$  models the diversity of the input distribution given a fixed class  $y_{\mathbf{x}}$  in a fixed task represented by the sample of  $\mathbf{z}$ . As before, we use (the decoder of) CVAE to implement DC X-Net. The training objective for synthetic task generation is then the empirical lower bound (Sohn et al., 2015) of VI on  $p(\omega | \mathbf{x}, y, \mathbf{z})$ :

$$\mathcal{J}_X \triangleq \sum_{t \in \mathcal{T}} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z} | \mathbf{y}, \mathcal{X}_t^s)} \left[ |\mathcal{X}_t|^{-1} \sum_{\mathbf{x} \in \mathcal{X}_t} (\mathbb{E}_{q_{\psi}(\omega | \mathbf{x}, y_{\mathbf{x}}, \mathbf{z})} [\log p_{\phi}(\mathbf{x} | y_{\mathbf{x}}, \mathbf{z}, \omega)] - D_{\text{KL}}[q_{\psi}(\omega | \mathbf{x}, y_{\mathbf{x}}, \mathbf{z}) \| p(\omega)]) \right]$$

where  $\phi$  is the parameter of the DC X-Net (decoder neural network),  $\psi$  is the parameter of the encoder neural network, and  $D_{KL}$  denotes the KL distance. In the training of DC X-Net, we sample one  $\mathbf{z}$  per update. We also sample one  $\omega$  per update to train with reparameterization tricks.

#### A.6 EVALUATION OF DISTANCE MEASURE BETWEEN TASKS FOR SETTINGS B TO E IN SEC. 4

For setting A, the subsampled Omniglot contains 15 characters, each of which has around 20 images. For settings B to E, the subsampled mini-Imagenet contains 50 classes. For each setting, we evaluate the distance measure between different types of tasks using *maximum mean discrepancy* (MMD) metric with radial basis function kernels on the  $\mathbf{z}$  samples, and show the results in Table 15. By comparing with Fig. 3, the distances evaluated using the MMD metric are in accordance with our visualization of the latent task embeddings.

Table 15: Values of MMD metric between different types of tasks for mini-Imagenet (settings B to E). Larger value means larger dissimilarity.

Setting B	brightness $-0.5$	normal	brightness $+0.5$
brightness $-0.5$	0	0.69	2.02
normal	0.69	0	1.26
brightness $+0.5$	2.02	1.26	0

Setting C	hue1 (red)	hue2 (green)	hue3 (blue)
hue1 (red)	0	0.70	1.97
hue2 (green)	0.70	0	1.69
hue3 (blue)	1.97	1.69	0

Setting D	original	3 $\times$ zoom-in	10 $\times$ zoom-in
original	0	0.58	2.06
3 $\times$ zoom-in	0.58	0	1.48
10 $\times$ zoom-in	2.06	1.48	0

Setting E	contrast=1/10	normal	contrast=10
contrast=1/10	0	0.73	0.95
normal	0.73	0	0.15
contrast=10	0.95	0.15	0

To illustrate that the latent task representation produced by IPML captures the semantic difference between tasks instead of the input-level modification, we consider the latent input representation learned by an ordinary classifier which has the same architecture as  $g_\theta$ . To compute the latent input embedding of a task, we average the last layer’s outputs over all data in this task. Fig. 6 shows a TSNE visualization of latent input embeddings from settings A to E based on the ordinary classifier. A comparison of Fig. 6 vs. Fig. 3 shows that except for setting C (i.e., different types of hue), the ordinary classifier cannot produce a latent input representation that captures the semantic difference between tasks. In contrast, IPML is capable of producing a latent task representation that captures such semantic difference between tasks.

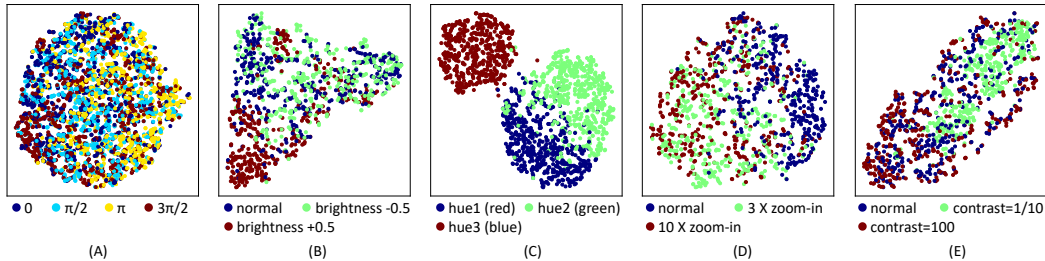


Figure 6: TSNE visualization of latent input embeddings from settings A to E based on an ordinary classifier.

#### A.7 EMPIRICAL EVALUATION OF TIME EFFICIENCY OF IPML ON THE ANONYMOUS E-COMMERCE COMPANY’S RISK DETECTION DATASET

This subsection empirically compares the time efficiency of different tested meta-learning algorithms on the aforementioned anonymous e-commerce company’s risk detection dataset which contains around 1 million entries. The training setting is the same as that in Appendix A.3 with a meta-batch size of 10. One training epoch spans approximately 950 iterations. Table 16 reports the results during meta-training: *Model forward* corresponds to the E step/SGHMC of IPML or the inner optimization loop of MAML-based algorithms, while *model backward* corresponds to the M step of IPML or the outer optimization loop of MAML-based algorithms. Our implementation of IPML obtains a total of 30 SGHMC samples per iteration for each task and MAML-based algorithms have 20 gradient steps in the inner optimization loop. The convergence is checked by inspecting the difference in the validation losses every 500 iterations.

The results in Table 16 show that the time efficiency of IPML is comparable to that of the first-order MAML baseline since (a) the forward passes are not that slow compared with the inner optimization loops of first-order MAML, and (b) the cost of backward passes of IPML is similar to that of first-order MAML. The time efficiency of BMAML baseline (Yoon et al., 2018) is lower than that of both IPML and vanilla MAML because BMAML maintains a set of particles (i.e., 5 particles in this setting), each representing a distinct network parameter. Unless efficient parallel optimization of those particles are implemented, BMAML or other particle-based Bayesian variant of MAML (e.g., (Jerfel et al., 2019)) can be a few times slower than vanilla MAML. In comparison, IPML does not maintain a set of particles of network weights and is thus more time-efficient.

Table 17 reports results during meta-testing (i.e., adaptation to a new task): *Model adaptation* corresponds to the E step/SGHMC of IPML or the inner optimization loop of MAML-based algorithms. Similar to the case during meta-training, IPML is comparable to first-order MAML in time efficiency during model adaptation, and BMAML has the worst time efficiency due to the use of particles. When performing prediction of new inputs, the time efficiency of all tested meta-learning algorithms are nearly the same. Both IPML and BMAML can estimate uncertainty by using more samples/particles in their predictions. For IPML, we have observed that using only 1 SGHMC sample in its prediction can already give satisfactory results.

The *neural process* (NP) uses amortized VI which introduces an additional encoder. Thus, it is faster in model forward but slower in model backward.

Table 16: Time efficiency of tested meta-learning algorithms during meta-training on the anonymous e-commerce company’s risk detection dataset.

	Total Iterations until Convergence	Model Forward	Model Backward
NP	8000	0.74s/iteration	0.14s/iteration
MAML (first-order)	8000	0.99s/iteration	0.024s/iteration
BMAML (5 particles)	7000	3.7s/iteration	0.045s/iteration
IPML	8000	1.24s/iteration	0.026s/iteration

Table 17: Time efficiency of tested meta-learning algorithms during meta-testing (i.e., adaptation to a new task) on the anonymous e-commerce company’s risk detection dataset.

	Support Set Size	Model Adaptation	Model Prediction (100 entries)
NP		0.098s	0.009s/sample
MAML (first-order)	100	0.112s	0.009s
BMAML (5 particles)		0.308s	0.009s/particle
IPML		0.132s	0.009s/SGHMC sample

#### A.8 ABLATION STUDY OF THE LIMITATIONS OF NEURAL PROCESS

In Sec. 4, we have empirically compared the performance of IPML with the *neural process* (NP) (Garnelo et al., 2018). For the implementation of NP, the default encoder is a 5-layer feed-forward neural network (i.e., with 100 hidden neurons) whose output is  $\mathbf{z}$ . The aggregator is the mean function, as proposed in the original work of Garnelo et al. (2018). The decoder has the same architecture as our IPML by adopting the same coupling of  $\mathbf{z}$  with the DNN  $g_\theta$  (i.e., by applying  $\mathbf{z}$  as a mask to the last DNN layer’s parameters). When performing classification tasks, we use the aforementioned robust-max likelihood (Appendix A.1).

We have observed that for both sinusoid regression (Table 1) and Omniglot (Table 2), NP performs unsatisfactorily as compared to IPML, likely due to the causes mentioned below.

**Optimization difficulties and overfitting.** NP performs amortized variational inference of  $\mathbf{z}$  through a heavily parameterized encoder which may introduce optimization difficulties and overfitting during meta-training. This is supported by the empirical evidence in Table 18: With an increasing depth of the encoder from the default of 2 layers, its performance improves and then deteriorates.

Table 18: *Mean square error* (MSE) on few-shot sinusoid regression.

Encoder depth	Sinusoid 5-shot	Sinusoid 10-shot
2 layers	0.780	0.417
3 layers	0.521	0.328
5 layers	0.460	0.264
7 layers	0.477	0.289

**No explicit modeling of  $\mathbf{x} \rightarrow y_{\mathbf{x}}$  relationship in the support set.** Our IPML algorithm performs SGHMC to obtain posterior samples of  $\mathbf{z}$  from  $p(\mathbf{z}|\mathbf{y}, \mathcal{X}_i^s)$ . SGHMC relies on our IP framework (Def. 1) which explicitly defines the  $\mathbf{x} \rightarrow y_{\mathbf{x}}$  relationship. Thus, the  $\mathbf{x} \rightarrow y_{\mathbf{x}}$  relationship in the support set is well captured by IPML with SGHMC. On the other hand, the encoder of NP takes in the simple concatenation of  $(\mathbf{x}, y_{\mathbf{x}})$  and thus does not explicitly capture the  $\mathbf{x} \rightarrow y_{\mathbf{x}}$  relationship in the support set, which we think is crucial to accurately recovering the true latent task posterior belief  $p(\mathbf{z}|\mathbf{y}, \mathcal{X}_i^s)$ .