A Appendix for Details of Deriving HTGM

A.1 The lower-bound of the likelihood function

In this section, we provide the details of the lower-bound in Eq. (3). By introducing the approximated posterior $q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau)$, the likelihood in Eq. (1) becomes (the superscript $*$ is neglected for clarity)

$$\begin{aligned}
&\ell(\mathcal{D}_\tau, \boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\log p_{\boldsymbol{\theta}}(\mathbf{e}_i|y_i) + \frac{1}{n}\sum_{i=1}^{n}\log\Big(\int_{\mathbf{v}_\tau} p(y_i|\mathbf{v}_\tau)p(\mathbf{v}_\tau)d\mathbf{v}_\tau\Big)\\
&= \frac{1}{n}\sum_{i=1}^{n}\log p_{\boldsymbol{\theta}}(\mathbf{e}_i|y_i) + \frac{1}{n}\sum_{i=1}^{n}\log\Big(\int_{\mathbf{v}_\tau} p(y_i|\mathbf{v}_\tau)p(\mathbf{v}_\tau)\frac{q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau)}{q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau)}d\mathbf{v}_\tau\Big)\\
&= \frac{1}{n}\sum_{i=1}^{n}\log p_{\boldsymbol{\theta}}(\mathbf{e}_i|y_i) + \frac{1}{n}\sum_{i=1}^{n}\log\Big(\int_{\mathbf{v}_\tau} q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau)\frac{p(y_i|\mathbf{v}_\tau)p(\mathbf{v}_\tau)}{q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau)}d\mathbf{v}_\tau\Big)\\
&\geq \frac{1}{n}\sum_{i=1}^{n}\log p_{\boldsymbol{\theta}}(\mathbf{e}_i|y_i) + \frac{1}{n}\sum_{i=1}^{n}\int_{\mathbf{v}_\tau} q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau)\Big[\log p(y_i|\mathbf{v}_\tau) + \log p(\mathbf{v}_\tau) - \log q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau)\Big]d\mathbf{v}_\tau\\
&= \frac{1}{n}\sum_{i=1}^{n}\log p_{\boldsymbol{\theta}}(\mathbf{e}_i|y_i) + \frac{1}{n}\sum_{i=1}^{n}\int_{\mathbf{v}_\tau} q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau)\Big[\log p(y_i|\mathbf{v}_\tau) + \log p(\mathbf{v}_\tau)\Big]d\mathbf{v}_\tau - \int_{\mathbf{v}_\tau} q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau)\log q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau)d\mathbf{v}_\tau\\
&= \frac{1}{n}\sum_{i=1}^{n}\log p_{\boldsymbol{\theta}}(\mathbf{e}_i|y_i) + \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{\mathbf{v}_\tau\sim q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau)}\Big[\log p(y_i|\mathbf{v}_\tau) + \log p(\mathbf{v}_\tau)\Big] + H(q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau))\\
&= \frac{1}{n}\sum_{i=1}^{n}\log p_{\boldsymbol{\theta}}(\mathbf{e}_i|y_i) + \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{\mathbf{v}_\tau\sim q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau)}\Big[\log p(y_i|\mathbf{v}_\tau) + \log\big(\sum_{z_\tau=1}^{r} p(\mathbf{v}_\tau|z_\tau)p(z_\tau)\big)\Big] + H(q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau))
\end{aligned} \tag{7}$$

where the fourth step uses Jensen's inequality. This completes the derivation of Eq. (3).

A.2 The upper-bound of the partition function

In Sec. 3.2, we apply an upper bound on the partition function in Eq. (2) for solving the challenging 2. The derivation of the upper bound is as follows.

$$\begin{aligned}
&\int_{\boldsymbol{\mu}^{\mathsf{c}}_{y_i}} \exp\big[-E_{\boldsymbol{\omega}}(\boldsymbol{\mu}^{\mathsf{c}}_{y_i};\mathbf{v}_\tau)\big]d\boldsymbol{\mu}^{\mathsf{c}}_{y_i} = \int_{\boldsymbol{\mu}^{\mathsf{c}}_{y_i}} \exp\big[-\min\big(\{||\boldsymbol{\mu}^{\mathsf{c}}_{y_i} - \mathbf{W}_j\mathbf{v}_\tau||_2^2\}_{j=1}^{N}\big)\big]d\boldsymbol{\mu}^{\mathsf{c}}_{y_i}\\
&= \int_{\boldsymbol{\mu}^{\mathsf{c}}_{y_i}} \max\Big(\big\{\exp\big[-||\boldsymbol{\mu}^{\mathsf{c}}_{y_i} - \mathbf{W}_j\mathbf{v}_\tau||_2^2\big]\big\}_{j=1}^{N}\Big)d\boldsymbol{\mu}^{\mathsf{c}}_{y_i} < \int_{\boldsymbol{\mu}^{\mathsf{c}}_{y_i}}\sum_{j=1}^{N}\exp\big[-||\boldsymbol{\mu}^{\mathsf{c}}_{y_i} - \mathbf{W}_j\mathbf{v}_\tau||_2^2\big]d\boldsymbol{\mu}^{\mathsf{c}}_{y_i}\\
&= \sum_{j=1}^{N}\int_{\boldsymbol{\mu}^{\mathsf{c}}_{y_i}} \exp\big[-||\boldsymbol{\mu}^{\mathsf{c}}_{y_i} - \mathbf{W}_j\mathbf{v}_\tau||_2^2\big]d\boldsymbol{\mu}^{\mathsf{c}}_{y_i} = N\sqrt{2^{d-1}\pi^d}
\end{aligned} \tag{8}$$

where the last equation is from the multidimensional Gaussian integral. This completes the derivation of the upper bound of the partition function.

A.3 The proof of Theorem 3.1

Proof. Let B_j denote a ball in $\mathbb{R}^d$. Its center is at $\mathbf{W}_j\mathbf{v}_\tau$ and its radius is $D_{hl}/3$. Because $\mathbf{W}_h\mathbf{v}_\tau$ and $\mathbf{W}_l\mathbf{v}_\tau$ $(1 \leq h, l \leq N)$ is the pair with the smallest Euclidean distance D_{hl}, for any pair of balls B_j and B_m we have $B_j \cap B_m = \varnothing$.

In other words, there is no overlap between any pair of balls. Therefore, if we compute the integral over the joint of all balls, we have

$$\int_{\boldsymbol{\mu}^{\mathsf{c}}_k \in \bigcup_{m=1}^{N} B_m} \exp\big[-E_{\boldsymbol{\omega}}(\boldsymbol{\mu}^{\mathsf{c}}_k;\mathbf{v}_\tau)\big]d\boldsymbol{\mu}^{\mathsf{c}}_k = \sum_{m=1}^{N}\int_{\boldsymbol{\mu}^{\mathsf{c}}_k \in B_m} \exp\big[-E_{\boldsymbol{\omega}}(\boldsymbol{\mu}^{\mathsf{c}}_k;\mathbf{v}_\tau)\big]d\boldsymbol{\mu}^{\mathsf{c}}_k \tag{9}$$

Also, because there is no overlap between any pair of balls, for each point $\boldsymbol{\mu}^{\mathsf{c}}_k \in B_m$, we have

$$-\min\Big(\{||\boldsymbol{\mu}^{\mathsf{c}}_k - \mathbf{W}_j\mathbf{v}_\tau||_2^2\}_{j=1}^{N}\Big) = -||\boldsymbol{\mu}^{\mathsf{c}}_k - \mathbf{W}_m\mathbf{v}_\tau||_2^2 \tag{10}$$

Therefore, we have the following derivation from Eq. (9).

$$\begin{aligned}\int_{\boldsymbol{\mu}_k^{\mathsf{c}} \in \bigcup_{m=1}^{N} B_m} \exp\big[-E_{\boldsymbol{\omega}}(\boldsymbol{\mu}_k^{\mathsf{c}};\mathbf{v}_\tau)\big] d\boldsymbol{\mu}_k^{\mathsf{c}} &= \sum_{m=1}^{N} \int_{\boldsymbol{\mu}_k^{\mathsf{c}} \in B_m} \exp\big[-E_{\boldsymbol{\omega}}(\boldsymbol{\mu}_k^{\mathsf{c}};\mathbf{v}_\tau)\big] d\boldsymbol{\mu}_k^{\mathsf{c}} \\ = \sum_{m=1}^{N} \int_{\boldsymbol{\mu}_k^{\mathsf{c}} \in B_m} \exp\big[-||\boldsymbol{\mu}_k^{\mathsf{c}} - \mathbf{W}_m \mathbf{v}_\tau||_2^2\big] d\boldsymbol{\mu}_k^{\mathsf{c}} &= N \int_{\boldsymbol{\mu}_k \in B_m} \exp\big[-||\boldsymbol{\mu}_k^{\mathsf{c}} - \mathbf{W}_m \mathbf{v}_\tau||_2^2\big] d\boldsymbol{\mu}_k^{\mathsf{c}}\end{aligned} \tag{11}$$

Meanwhile, since $\bigcup_{m=1}^{N} B_m$ is a sub-area of the entire $\mathbb{R}^d$ space, we have

$$\int_{\boldsymbol{\mu}_k^{\mathsf{c}} \in \bigcup_{m=1}^{N} B_m} \exp\big[-E_{\boldsymbol{\omega}}(\boldsymbol{\mu}_k^{\mathsf{c}};\mathbf{v}_\tau)\big] d\boldsymbol{\mu}_k^{\mathsf{c}} \leq \int_{\boldsymbol{\mu}_k^{\mathsf{c}}} \exp\big[-E_{\boldsymbol{\omega}}(\boldsymbol{\mu}_k^{\mathsf{c}};\mathbf{v}_\tau)\big] d\boldsymbol{\mu}_k^{\mathsf{c}} \tag{12}$$

According to the multidimensional Gaussian integral, we have

$$\lim_{D_{hl}\to\infty} \int_{\boldsymbol{\mu}_k^{\mathsf{c}} \in B_m} \exp\big[-E_{\boldsymbol{\omega}}(\boldsymbol{\mu}_k^{\mathsf{c}};\mathbf{v}_\tau)\big] d\boldsymbol{\mu}_k^{\mathsf{c}} = \sqrt{2^{d-1}\pi^d} \tag{13}$$

Therefore,

$$\lim_{D_{hl}\to\infty} \int_{\boldsymbol{\mu}_k^{\mathsf{c}}} \exp\big[-E_{\boldsymbol{\omega}}(\boldsymbol{\mu}_k^{\mathsf{c}};\mathbf{v}_\tau)\big] d\boldsymbol{\mu}_k^{\mathsf{c}} \geq N\sqrt{2^{d-1}\pi^d} \tag{14}$$

Since $N\sqrt{2^{d-1}\pi^d}$ is its upper bound, based on the squeeze theorem, we have

$$\lim_{D_{hl}\to\infty} \int_{\boldsymbol{\mu}_k^{\mathsf{c}}} \exp\big[-E_{\boldsymbol{\omega}}(\boldsymbol{\mu}_k^{\mathsf{c}};\mathbf{v}_\tau)\big] d\boldsymbol{\mu}_k^{\mathsf{c}} = N\sqrt{2^{d-1}\pi^d} \tag{15}$$

which completes the proof of Theorem 3.1. □

A.4 The training algorithm of HTGM

The training algorithm of HTGM is summarized in Algorithm 1.

B Appendix for Further Discussion

B.1 Discussion about the novel task discussion and meta-learning

As we discussed in Sec. 2, to the best of our knowledge, our proposed method HTGMis the first work that jointly consider the task mixture distribution and novel task detection in meta-testing stage. There are some works considering how to identify novel task clusters in meta-training stage based on task embedding [47] or task likelihood [11]. However, they have their own respective drawbacks when handling novel task detection in meta-testing stage. For task-embedding-based method like [47], it does not explicitly model the task distribution. Instead, it considers how to model the task membership of the learnt clusters. As a result, they can only identify the outlying task clusters rather than individual novel tasks. However, in meta-testing stage, we expect the model to identify each individual novel task and raise alerts. The task-likelihood-based method DPMM [11] can handle individual novel tasks. However, it is hard for them to simultaneously handle quick detection and adaptation. This is because its likelihood was built on the entire model parameters, leading to model-dependent and time consuming computation. It is not a big issue for meta-training, but will serious limit its application to streaming tasks in meta-testing (e.g., in auto-driving domain) where efficiency is critical for timely alarms of novel tasks.

B.2 Discussion about the relationship between HTGM and HGM model

To the best of our knowledge, the Hierarchical Gaussian Mixture (HGM) model has appeared in the traditional works [8, 30, 3] for hierarchical clustering by applying Gaussian Mixture model agglomeratively or divisively on the input samples. They are unsupervised methods that infer clusters of samples, but do not pre-train embedding models (or parameter initializations) that could be fine-tuned for the adaptation to new tasks in meta-learning. Therefore, these methods are remarkably different from meta-learning methods, and we think it is a non-trivial problem to adapt the concept of HGM to

Algorithm 1: Hierarchical Gaussian Mixture based Task Generative Model (HTGM)

Input: encoder $f_{\boldsymbol{\theta}}$, training dataset $\mathcal{D}^{\text{tr}}$, hyperparameters r, σ, $\bar{\sigma}$
Output: model parameters $\{\boldsymbol{\theta}, \boldsymbol{\omega}\}$

Pre-train the encoder $f_{\boldsymbol{\theta}}$ via ProtoNet with augmentations.
Pre-train the energy function in Eq. (2) by maximizing $\frac{1}{n}\sum_{i=1}^{n}\log p_{\boldsymbol{\theta},\boldsymbol{\omega}}(\mathbf{e}_i|y_i) + \log p_{\omega}(y_i|\mathbf{v}_\tau)$
for $i \leftarrow 1$ *to MaxEpoch* **do**
  /* E-step */
  $\mathcal{V} = \varnothing$
  for $\{\mathcal{D}^{s}_{\tau} = \{(\mathbf{x}^{s}_i, y^{s}_i)\}_{i=1}^{ns}, \mathcal{D}^{q}_{\tau} = \{(\mathbf{x}^{q}_i, y^{q}_i)\}_{i=1}^{nq}\}$ *in Dataloader($\mathcal{D}^{tr}$)* **do**
    /* load a task episode */
    $\{\mathbf{e}^{\mathsf{s}}_i\}_{i=1}^{n\mathsf{s}} = \{f_{\boldsymbol{\theta}}(\mathbf{x}^{\mathsf{s}}_i)\}_{i=1}^{n\mathsf{s}}$; // embeddings of the support set
    $\boldsymbol{\mu}^{\mathsf{a}}_{z_\tau}$ = Task-Pooling(Class-Pooling($\{(\mathbf{e}^{\mathsf{s}}_i, y^{\mathsf{s}}_i)\}_{i=1}^{n\mathsf{s}}$)) ; // the mean of $q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau)$
    Sample a task embedding $\mathbf{v}_\tau$ from $q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau) = \mathcal{N}(\boldsymbol{\mu}^{\mathsf{a}}_{z_\tau}, \bar{\sigma}^2\mathbf{I})$
    $\mathcal{V} = \mathcal{V} \cup \{\mathbf{v}_\tau\}$
  end
  $\{z_\tau\}_{\tau=1}^{|\mathcal{V}|}$, $\{\boldsymbol{\mu}^{\mathsf{t}}_1, ..., \boldsymbol{\mu}^{\mathsf{t}}_r, \boldsymbol{\Sigma}^{\mathsf{t}}_1, ..., \boldsymbol{\Sigma}^{\mathsf{t}}_r\}$ = GMM($\mathcal{V}$). ; // fit a GMM to $\mathcal{V}$, where $\{z_\tau\}_{\tau=1}^{|\mathcal{V}|}$ represents the labeling of the $\mathbf{v}_\tau$'s in $\mathcal{V}$
  /* M-step */
  for $\{\mathcal{D}^{s}_{\tau} = \{(\mathbf{x}^{s}_i, y^{s}_i)\}_{i=1}^{ns}, \mathcal{D}^{q}_{\tau} = \{(\mathbf{x}^{q}_i, y^{q}_i)\}_{i=1}^{nq}\}$ *in Dataloader($\mathcal{D}^{tr}$)* **do**
    /* load a task episode */
    $\{\mathbf{e}^{\mathsf{s}}_i\}_{i=1}^{n\mathsf{s}} = \{f_{\boldsymbol{\theta}}(\mathbf{x}^{\mathsf{s}}_i)\}_{i=1}^{n\mathsf{s}}$; // forward pass
    $\{\mathbf{e}^{\mathsf{q}}_i\}_{i=1}^{n\mathsf{q}} = \{f_{\boldsymbol{\theta}}(\mathbf{x}^{\mathsf{q}}_i)\}_{i=1}^{n\mathsf{q}}$; // forward pass
    $\{\boldsymbol{\mu}^{\mathsf{c}}_1, ..., \boldsymbol{\mu}^{\mathsf{c}}_N\}^{\mathsf{s}}$ = Class-Pooling($\{(\mathbf{e}^{\mathsf{s}}_i, y^{\mathsf{s}}_i)\}_{i=1}^{n\mathsf{s}}$)
    $\boldsymbol{\mu}^{\mathsf{a}}_{z_\tau}$ = Task-Pooling($\{\boldsymbol{\mu}^{\mathsf{c}}_1, ..., \boldsymbol{\mu}^{\mathsf{c}}_N\}^{\mathsf{s}}$) ; // the mean of $q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau)$
    Sample a task embedding $\mathbf{v}_\tau$ from $q_{\boldsymbol{\phi}}(\mathbf{v}_\tau|\mathcal{D}^{\mathsf{s}}_\tau) = \mathcal{N}(\boldsymbol{\mu}^{\mathsf{a}}_{z_\tau}, \bar{\sigma}^2\mathbf{I})$
    for *j = 1, ..., N* **do**
      $\bar{\boldsymbol{\mu}}^{\mathsf{c}}_j = \alpha\boldsymbol{\mu}^{\mathsf{c}}_j + (1-\alpha)\mathbf{W}_{l^*}\mathbf{v}_{\tau'}$ where $l^* = \arg\min_{1\le l\le N} D(\boldsymbol{\mu}^{\mathsf{c}}_j, \mathbf{W}_l\mathbf{v}_{\tau'})$
    end
    Calculate $\ell(\{\mathbf{e}^{\mathsf{q}}_i\}_{i=1}^{n\mathsf{q}}, \mathcal{V}, \{\bar{\boldsymbol{\mu}}^{\mathsf{c}}_j\}_{j=1}^{N}, \{\boldsymbol{\mu}^{\mathsf{t}}_1, ..., \boldsymbol{\mu}^{\mathsf{t}}_r, \boldsymbol{\Sigma}^{\mathsf{t}}_1, ..., \boldsymbol{\Sigma}^{\mathsf{t}}_r\}, \sigma, \boldsymbol{\omega})$; // calculate the loss in Eq. (5) using Eq. (3) and Eq. (4)
    $\boldsymbol{\theta}, \boldsymbol{\omega}$ = SGD($\ell, \boldsymbol{\theta}, \boldsymbol{\omega}$) ; // update model parameters
  end
end

solve the meta-learning problem. To this end, we need to (1) identify the motivation; and (2) solve the new technical challenges. For (1), we found the hierarchical structure of mixture distributions naturally appears when we want to model the generative process of tasks from a mixture of distributions, where each task contains another mixture distribution of classes (as suggested by Eq. (1)). In other words, the motivating point of our method is more on meta-learning than HGM. However, we think drawing such a connection between meta-learning and HGM is a novel contribution. For (2), our method is different from traditional HGM in (a) its generative process of tasks (Sec. 3.1), which is a theoretical extension of the widely used empirical process of generating tasks in meta-learning; (b) its Gibbs-style task-conditional distribution (Eq. (2)) for fitting uniformly sampled classes; (c) the metric-based end-to-end meta-learning framework (Fig. 1) (note the traditional HGM is not for learning embeddings); (d) the non-trivial derivation of the optimization algorithm in Sect. 3.2 and Alg. 1; and (e) the novel model adaptation process in Sec. 3.3. Solving the technical challenges in the new generative model is another novel contribution of the proposed method.

B.3 Discussion about the related multi-task learning methods

The modeling of the clustering/grouping structure of tasks or the mixture of distributions of tasks has been studied in multi-tasking learning (MTL). In [46, 9], tasks are assumed to have a clustering structure, and the model parameters of the tasks in the same cluster are drawn to each other via optimization on their L2 distances. In [13], a subspace based regularization framework was proposed for grouping task-specific model parameters, where the tasks in the same group are assumed to lie in the same low dimensional subspace for parameter sharing. The method in [16] also uses the subspace based sharing of task parameters, but allows two tasks from different groups to overlap by having one or more bases in common. The method in [32] introduces a generative model for task-specific model parameters that encourages parameter sharing by modeling the latent mixture distribution of the parameters via the Dirichlet process and Beta process.

The key difference between these methods and our method HTGM lies in the difference between MTL and meta-learning. In an MTL method, all tasks are known *a priori*, *i.e.*, the testing tasks are from the set of training tasks, and the model is non-inductive at the task-level (but it is inductive at the sample-level). In HTGM, testing tasks can be disjoint from the set of training tasks, thus the model is inductive at the task-level. In particular, we aim to allow testing tasks that are not from the distribution of the training tasks by enabling the detection of novel tasks, which is an extension of the task-level inductive model. The second difference lies in the generative process. The method in [32] models the generative process of the task-specific model parameters (*e.g.*, the weights in a regressor). In contrast, HTGM models the generative process of each task by generating the classes in it, and the samples in the classes hierarchically, *i.e.*, the $(\mathbf{x}, y)$'s (in Eq. (1) and Sec. 3.1). In this process, we allow our model to fit uniformly sampled classes given a task (without specifying a prior on the distance function on classes) by the proposed Gibbs distribution in Eq. (2). Other remarkable differences to the aforementioned MTL methods include the inference network (Fig. 1(b)), which allows the inductive inference on task embeddings and class prototypes; the optimization algorithm (Sec. 3.2) to our specific loss function in Eq. (3), which is from the likelihood in Eq. (1); and the model adaptation algorithm (Sec. 3.3) for performing predictions in a testing task, and detecting novel tasks. As such, the MTL methods can not be trivially applied to solve our problem.

B.4 Further interpretation of the task-conditional distribution

The task-conditional class distribution $p_{\boldsymbol{\omega}}(y_i = k|\mathbf{v}_\tau)$ in Eq. (2) is defined through an energy function $E_{\boldsymbol{\omega}}(\boldsymbol{\mu}^{\mathsf{c}}_k; \mathbf{v}_\tau) = \min(\{||\boldsymbol{\mu}^{\mathsf{c}}_k - \mathbf{W}_j\mathbf{v}_\tau||_2^2\}_{j=1}^N)$ with trainable parameters $\boldsymbol{\omega} = \{\mathbf{W}_1, ..., \mathbf{W}_N\}$, for allowing uniformly sampled classes per task. The conditional distribution $p(y_i|\mathbf{v}_\tau)$ represents how classes distribute for a given task τ. The reason for its definition in Eq. (2) is as follows. If it is a Gaussian distribution with $\mathbf{v}_\tau$ (*i.e.*, task embedding) as the mean, $p(y_i = k|\mathbf{v}_\tau)$ can be interpreted as the density at the representation of the k-th class in this Gaussian distribution, *i.e.*, the density at $\boldsymbol{\mu}_k$, which is the mean/surrogate embedding of the k-th class. One problem of this Gaussian $p(y_i|\mathbf{v}_\tau)$ is that different classes, *i.e.*, different $\boldsymbol{\mu}_{y_i}$'s, are not uniformly distributed, contradicting the practice that given a dataset (e.g., images), classes are often uniformly sampled for constituting a task in the empirical studies. Using a uniformly sampled set of classes to fit the Gaussian distribution $p(y_i|v_\tau)$ will lead to an ill-posed learning problem, as described in Sec. 3.1. To solve it, we introduced $\boldsymbol{\omega} = \{\mathbf{W}_1, ..., \mathbf{W}_N\}$ in the energy function $E_{\boldsymbol{\omega}}(\boldsymbol{\mu}^{\mathsf{c}}_k; \mathbf{v}_\tau)$ in Eq. (2). $\mathbf{W}_j \in \mathbb{R}^{d\times d}$ $(1 \le j \le N)$ can be interpreted as projecting $\mathbf{v}_\tau$ to the j-th space spanned by the basis (*i.e.*, columns) of $\mathbf{W}_j$. There are N different spaces for $j = 1, ..., N$. Thus, the N projected task means $\mathbf{W}_1\mathbf{v}_\tau, ..., \mathbf{W}_N\mathbf{v}_\tau$ are in N different spaces. Fitting the energy function $E_{\boldsymbol{\omega}}(\boldsymbol{\mu}^{\mathsf{c}}_k; \mathbf{v}_\tau)$ to N uniformly sampled classes $\boldsymbol{\mu}^{\mathsf{c}}_1, ..., \boldsymbol{\mu}^{\mathsf{c}}_N$, which tend to be far from each other because they are uniformly random, tends to learn $\mathbf{W}_1, ..., \mathbf{W}_N$ that project $\mathbf{v}_\tau$ to N far apart spaces that fit each of the $\boldsymbol{\mu}^{\mathsf{c}}_1, ..., \boldsymbol{\mu}^{\mathsf{c}}_N$ by closeness, due to the min-pooling operation. This mitigates the aforementioned ill-posed learning problem.

C Appendix for Implementation Details

C.1 The setup of the compared models

Encoder of Metric-based Meta-Learning. For fairness, for all metric-based methods, including ProtoNet [39], MetaOptNet [22], ProtoNet-Aug [40], FEATS [49] and NCA [17], following [41, 22], we apply ResNet-12 as the encoder. ResNet-12 has 4 residual blocks, each has 3 convolutional layers with a kernel size of 3×3. ResNet-12 uses dropblock as a regularizer, and its number of filters is (60, 160, 320, 640). For MetaOptNet, following its paper [22], we flattened the output of the last convolutional layer to acquire a 16000-dimensional feature as the image embedding. For other baselines, following [41], we used a global average-pooling layer on the top of the last residual block to acquire a 640-dimensional feature as the image embedding.

Further Details. Following [39], ProtoNet, ProtoNet-Aug, and NCA use Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.99$. We did grid-search for the initial learning rate of the Adam within

$\{1e^{-2}, 1e^{-3}, 1e^{-4}\}$, where $1e^{-3}$ was selected, which is the same as the official implementation provided by the authors. For FEATS, we chose transformer as the set-to-set function based on the results reported by [49]. When pre-training the encoder in FEATS, following its paper [49], we applied the same setting as ProtoNet, which is to use Adam optimizer with an initial learning rate of $1e^{-3}$, $\beta_1 = 0.9$ and $\beta_2 = 0.99$. When training its aggregation function, we grid-searched the initial learning rate in $\{1e^{-4}, 5e^{-4}, 1e^{-5}\}$ since a larger learning rate leads to invalid results on our datasets. The optimal choice is $1e^{-4}$. For MetaOptNet, following its paper [22], we used SGD with Nesterov momentum of 0.9, an initial learning rate of 0.1 and a scheduler to optimize it, and applied the quadratic programming solver OptNet [2] for the SVM solution in it.

C.2 The details of the setup for novel task detection

In the experiments on novel task detection in Sec. 4.1, the number of in-distribution tasks (from the Original domain) in the test set is 4000 (1000 per task cluster) and the number of novel tasks (from the Blur and Pencil domains) in the test set is 8000 (4000 for the Blur and 4000 for the Pencil).

D Appendix for Experimental Results

D.1 Analysis of σ

Setting of σ	Bird	Texture	Aircraft	Fungi
0.1	69.33	46.92	75.20	50.78
0.5	70.00	**47.98**	75.38	**52.38**
1.0 (Ours)	**70.12**	47.76	**75.52**	52.06
10.0	69.4	47.28	75.32	51.5

Table 4: Analysis of different σ

Tabel 4 report the effect of different σ on the classification performance (5-way-1-shot classification on Multi-Plain dataset). As shown in the table, although the too low or too high setting of this hyper-parameter will hurt the performance, in general the model is robust toward the setting of σ.

D.2 Analysis of $\bar{\sigma}$

Setting of $\bar{\sigma}$	Bird	Texture	Aircraft	Fungi
0.05	69.78	**48.36**	74.36	51.34
0.1(Ours)	**70.12**	47.76	**75.52**	**52.06**
0.2	70.02	47.50	75.30	51.74
0.5	69.02	46.66	74.46	51.00

Table 5: Analysis of different $\bar{\sigma}$

Tabel 5 summarize how different $\bar{\sigma}$ influence classification performance (5-way-1-shot classification on Multi-Plain dataset). In general, different settings of $\bar{\sigma}$ will influence the model performance at a marginal level, indicating our model's robustness toward this hyper-parameter.

D.3 Impact of GMM component number

Number of components r	2	4	8	16	32
Silhouette score	47.70	57.61	12.76	7.81	6.19

Table 6: Analysis on the number of mixture components

Different choices of the number of mixture components does not significantly influence the model classification performance. However, the clustering quality may vary due to the different numbers of components. Here, we report the Silhouette score [36, 37] *w.r.t.* the number in Table 6. From Table 6, we can see that selecting a component number close to the ground-truth component number of the distribution can benefit the clustering quality.

D.4 Classification performance of the ablation variants

Ablation Variants	Bird	Texture	Aircraft	Fungi
HTGM w/o GMM	68.86	**48.00**	**75.74**	**52.28**
HTGM-Gaussian	69.52	47.3	75.38	51.34
HTGM	**70.12**	47.76	75.52	52.06

Table 7: Ablation study of different variants of our proposed method.

We summarize the classification performance of the two Ablation Variants HTGM w/o GMM and HTGM-Gaussian in Table 7. As we can see, our unique designs improve the novel task detection performance without significantly decreasing the classification performance.

D.5 Ablation analysis of optimization-based methods

Setting	Model	Bird	Texture	Aircraft	Fungi	Average
5-way-1-shot	ANIL-MAML	62.64±0.90	43.86±0.78	70.03±0.85	48.34±0.89	56.22
	ANIL-HSML	64.33±0.87	43.77±0.79	69.71±0.84	47.75±0.89	56.39
	ANIL-ARML	65.98±0.87	43.57±0.78	70.28±0.84	48.48±0.92	57.08
	HTGM (ours)	**70.12±1.28**	**47.76±1.49**	**75.52±1.24**	**52.06±1.41**	**61.37**
5-way-5-shot	ANIL-MAML	74.38±0.73	55.36±0.74	79.78±0.63	59.57±0.79	67.27
	ANIL-HSML	78.18±0.71	57.70±0.75	81.32±0.62	59.83±0.81	69.26
	ANIL-ARML	78.79±0.71	57.61±0.73	81.86±0.59	60.19±0.81	69.61
	HTGM (ours)	**82.27±0.74**	**60.67±0.78**	**88.48±0.52**	**65.70±0.79**	**74.28**

Table 8: More results (accuracy±95% confidence) of the optimization-based methods.

We selected the two best performed optimization-based baselines HSML and ARML, and the widely used method MAML for this ablation analysis. Table 8 summarizes the performance of MAML, HSML and ARML trained in ANIL method [34], *i.e.*, we pre-trained the ResNet-12 by ProtoNet, froze the encoder, and fine-tuned the last fully-connected layers using MAML, HSML and ARML on Plain-Multi dataset. From Table 8, the performance of ANIL-MAML is better than MAML in Table 1, similar to the observation in [34], indicating the effectiveness of ANIL method. However, ANIL-HSML and ANIL-ARML perform similarly to ANIL-MAML, losing their superiority of modeling the mixture distribution of tasks achieved when implemented without ANIL as in Table 1 (up to 5.6% average improvement). This is because the cluster layer in HSML and the graph layer in ARML both affect the embeddings learned through backpropagation, *i.e.*, they were designed for joint training with the encoder. When the encoder is frozen, they cannot work properly. For this reason, to be consistent with the existing research [47, 48] that demonstrated the difference between HSML/ARML and MAML, we used their original designs in Sec. 4. Meanwhile, we observed the proposed HTGM outperforms MAML, HSML, and ARML trained in ANIL method, this is because MAML cannot model the mixture distribution of tasks, while HSML and ARML cannot work properly when trained in ANIL method.

D.6 More results on the Mini-ImageNet dataset

Model	5-way-1-shot	5-way-5-shot
ProtoNet-Aug	59.40±0.93	74.68±0.45
HTGM (ours)	**61.80±0.95**	**74.55±0.45**

Table 9: Comparison of the proposed method and ProtoNet-Aug on the Mini-ImageNet dataset.

In the case when the task distribution is not a mixture, our model would degenerate to and perform similarly to the general metric-based meta-learning methods, *e.g.*, ProtoNet, which only considers a uni-component distribution. To confirm this, we added an experiment that compares our model with ProtoNet-Aug on Mini-ImageNet [43], which does not have the same explicit mixture distributions as in the Plain-Multi and Art-Multi datasets in Section 4. The results are summarized in Table 9. From the table, we observe our method performs comparably to ProtoNet, which validates the aforementioned guess. Meanwhile, together with the results in Table 1 and Table 2, the proposed method could be considered as a generalization of the metric-based methods to the mixture of task distributions.