Contents

A Derivation for EBML

A.1 Derivation for Training Objective in Eqn. (8)

We start the derivation from maximizing the ELBO (Eqn. (5)) for a single training task $\mathcal{T}_i$ w.r.t. the parameters of the EBMs and the task latent posterior distribution, i.e., λ, ω and ψ,

$$\arg\max_{\lambda,\psi,\omega} \log\ p(\mathcal{T}_i) \xrightarrow{\text{Eqn. (5)}} \arg\max_{\lambda,\psi,\omega} \underbrace{\mathbb{E}_{\phi_i\sim q_\psi(\phi_i|\mathcal{T}_i^s)}\left[\log p_\omega(\mathbf{X}_i,\mathbf{Y}_i|\phi_i)\right]}_{\textit{(I)}} - \mathrm{KL}(q_\psi(\phi_i|\mathcal{T}_i^s)||p_\lambda(\phi_i)) \tag{12}$$

where q_ψ, p_ω and p_λ denote the model distributions of the latent posterior, the data EBM and the prior EBM, respectively. Recall p_λ and p_ω are EBMs where $p_\omega(\mathbf{X}_i,\mathbf{Y}_i \mid \phi_i) = \prod_j \frac{\exp\ (-E_\omega(\mathbf{x}_{ij},y_{ij},\phi_i))}{Z(\omega,\phi_i)}$ and $p_\lambda = \frac{\exp\ (-E_\lambda(\phi_i))}{Z(\lambda)}$. We rewrite the KL term as

$$\begin{aligned}
-\mathrm{KL}(q_\psi(\phi_i|\mathcal{T}_i^s)||p_\lambda(\phi_i)) &= -\,\mathbb{E}_{q_\psi(\phi_i|\mathcal{T}_i^s)}\Big[\log\frac{q_\psi(\phi_i|\mathcal{T}_i^s)}{p_\lambda(\phi_i)}\Big] &(13)\\
&= -\,\mathbb{E}_{q_\psi(\phi_i|\mathcal{T}_i^s)}\Big[\log q_\psi(\phi_i|\mathcal{T}_i^s\Big] - \mathbb{E}_{q_\psi(\phi_i|\mathcal{T}_i^s)}\Big[\frac{1}{\log\ p_\lambda(\phi_i)}\Big] &(14)\\
&= \underbrace{\mathcal{H}(q_\psi(\phi_i|\mathcal{T}_i^s)}_{\textit{(III)}} + \underbrace{\mathbb{E}_{q_\psi(\phi_i|\mathcal{T}_i^s)}[\log\ p_\lambda(\phi_i)]}_{\textit{(II)}} &(15)
\end{aligned}$$

The two log-likelihood terms for EBMs, *(I)* and *(II)*, can be rewritten using the learning gradient in Eqn. (4), i.e.,

$$\begin{aligned}
\mathbb{E}_{\phi_i\sim q_\psi(\phi_i|\mathcal{T}_i^s)}\left[\log p_\omega(\mathbf{X}_i,\mathbf{Y}_i|\phi_i)\right] &\xrightarrow{\text{Eqn. (4)}} \mathbb{E}_{\boldsymbol{\phi}_i\sim q_\psi(\boldsymbol{\phi}_i|\mathcal{T}_i^s)}\Bigg[\sum_j^{N_i} -E_\omega(\mathbf{x}_{ij},y_{ij},\boldsymbol{\phi}_i)\\
&\qquad + \mathbb{E}_{p_\omega(\mathbf{x}',y'|\phi_i)}[E_\omega(\mathbf{x}'_{ij},y'_{ij},\boldsymbol{\phi}_i)]\Bigg] &(16)\\
\mathbb{E}_{q_\psi(\phi_i|\mathcal{T}_i^s)}[\log\ p_\lambda(\phi_i)] &\xrightarrow{\text{Eqn. (4)}} \mathbb{E}_{q_\psi(\boldsymbol{\phi}_i|\mathcal{T}_i^s)}\Big[-E_\lambda(\boldsymbol{\phi}_i) + \mathbb{E}_{p_\lambda(\boldsymbol{\phi}'_i)}[E_\lambda(\boldsymbol{\phi}'_i)]\Big]. &(17)
\end{aligned}$$

Combining with the entropy term in *(III)* and take the expectation w.r.t. the training task distributing $\mathcal{P}_{ID}$ we have our training objective in Eqn. (8).

A.2 Derivation for Energy Sum in Eqn. (9)

The log-likelihood of a task $\mathcal{T}_i$ writes

$$\begin{aligned}
\log\ p(\mathcal{T}_i) &= \log \int \prod_{j=1}^{N_i} p(\mathbf{x}_{ij},y_{ij}|\phi_i)p_\lambda(\phi_i)\ d\phi_i. &(18)\\
&\geq\ \mathbb{E}_{\phi_i\sim q_\psi(\phi_i|\mathcal{T}_i^s)}\left[\log p_\omega(\mathbf{X}_i,\mathbf{Y}_i|\phi_i)\right] - \mathrm{KL}(q_\psi(\phi_i|\mathcal{T}_i^s)||p_\lambda(\phi_i)) &(19)\\
&= \mathbb{E}_{q_\psi(\phi_i|\mathcal{T}_i^s)}\big[\underbrace{\log p_\omega(\mathbf{X}_i,\mathbf{Y}_i|\phi_i)}_{\propto\ -E_\omega(\mathbf{X}_i,\mathbf{Y}_i,\phi_i)}\big] + \mathbb{E}_{q_\psi(\phi_i|\mathcal{T}_i^s)}\big[\underbrace{\log\ p_\lambda(\phi_i)}_{\propto\ -E_\lambda(\phi_i)}\big] + \mathcal{H}(q_\psi(\phi_i|\mathcal{T}_i^s)\big], &(20)
\end{aligned}$$

which is lower-bounded by the ELBO in Eqn. (20) characterized by the learned q_ψ, p_ω and p_λ. The ELBO is proportional to the sum of two energy functions and the entropy of the posterior distribution q_ψ, all of which can be easily calculated via feed-forward passes of the training samples.

Since the majority of the state-of-the-art meta-learning algorithms (including CNP [8] and SimpleCNAPS [1]) adopt the MAP estimation of the posterior q_ψ which is determinisitc, the entropy essential becomes zero, and the expectations $\mathbb{E}_{q_\psi(\phi_i|\mathcal{T}_i^s)}$ in the first and second terms in Eqn. (20) simplify to energy function evaluation at $\mathbf{X}_i, \mathbf{Y}_i$ and ϕ_i, respectively. Finally, we negate Eqn. (20) so that the OOD scores for in-distribution tasks are lower than that of the out-of-distribution ones.

A.3 Derivation for Eqn. (11) as an Approximation to Bayesian Posterior Inference

Given a new test task $\mathcal{T}_i$, Bayesian inference aims to infuse the meta-learned prior knowledge with the observed support set $\mathbf{X}_i^s, \mathbf{Y}_i^s$ for inferring a small set of unknown task-specific parameters ζ. This is akin to maximizing the log-likelihood of the support set w.r.t. the task-specific parameters ζ which defines the posterior latent distribution $q_{\psi\cup\zeta}(\phi_i \mid \mathcal{T}_i^s)$ under the regularization of a prior. First, the tractable ELBO for the prior predictive likelihood is

$$\log\ p_{\lambda,\psi,\omega}(\mathcal{T}_i^s) = \log \int \prod_{j=1}^{N_i^s} p_\omega(\mathbf{x}_{ij}, y_{ij}|\phi_i) p_\lambda(\phi_i)\ d\phi_i. \tag{21}$$

$$\geq\ \mathbb{E}_{q_\psi(\phi_i|\mathcal{T}_i^s)}\ [\sum_{j=1}^{N_i^s} \log p_\omega(\mathbf{x}_{ij}, y_{ij}|\phi_i)] - \mathrm{KL}(q_\psi(\phi_i|\mathcal{T}_i^s)||p_\lambda(\phi_i)). \tag{22}$$

Next, we introduce the task-specific parameter ζ in the latent posterior and formulate the Bayesian posterior inference objective as

$$\arg\min_{\zeta} \mathbb{E}_{q_{\psi\cup\zeta}(\phi_i|\mathcal{T}_i^s)}\ [-\sum_{j=1}^{N_i^s} \log p_\omega(\mathbf{x}_{ij}, y_{ij}|\phi_i)] + \mathrm{KL}(q_{\psi\cup\zeta}(\phi_i|\mathcal{T}_i^s)||p_\lambda(\phi_i)). \tag{23}$$

Assuming a maximum a posterior (MAP) estimate of the task-specific latent distribution which approximates $q_{\psi\cup\zeta}(\phi_i|\mathcal{T}_i^s)$ by a Dirac-delta function $q_\psi(\phi_i|\mathcal{T}_i^s) = \delta(\phi_i - \hat{\phi}_i \mid \mathcal{T}_i^s)$. As a result, the second KL term reduces to a likelihood evaluation, i.e., $\mathrm{KL}(q_\psi(\phi_i|\mathcal{T}_i^s)||p_\lambda(\phi_i)) = \mathbb{E}_{q_\psi(\phi_i|\mathcal{T}_i^s)}[\log\frac{q_\psi(\phi_i|\mathcal{T}_i^s)}{p_\lambda(\phi_i)}] = -\log\ p_\lambda(\phi_i) = E_\lambda(\phi_i) + \log\ Z(\lambda)$. Since the (log-)partition function $\log\ Z(\lambda)$ is a constant w.r.t. the $\arg\min$ parameter ζ, we then have $\arg\min_\zeta \mathrm{KL}(q_\psi(\phi_i|\mathcal{T}_i^s)||p_\lambda(\phi_i)) = \arg\min_\zeta E_\lambda(\phi_i)$. From here, we see that minimizing the task prior energy approximates the minimization of the KL-divergence between the task-specific posterior $q_{\psi\cup\zeta}$ and the meta-learned ID prior p_λ in the Bayesian posterior inference objective, thus it acts as a meta-regularizer to combat over-fitting in adaptation. In practice, we found that using a margin loss for this prior energy minimization, i.e., $\arg\min_\zeta \max(E_\lambda(\phi_i) - m, 0)$, can yield better empirical performance.

While for the first log-likelihood term inside the expectation, $-\sum_{j=1}^{N_i^s} \log p_\omega(\mathbf{x}_{ij}, y_{ij}|\phi_i)$ which is equivalent to the sum of data energy scores $\sum_{j=1}^{N_i^s}[E_\omega(\mathbf{x}_{ij}, y_{ij}, \phi_i) + \log\ Z(\omega, \phi_i)]$, we use the decoder ω_2 with the task-specific prediction loss i.e., cross-entropy loss, in the base meta-learning algorithm as a tractable surrogate, as discussed in implementation of the Experiment section 5 and also in Appendix B.4.1. This thus maintains the data-level predictive ability of model during adaptation.

B Experiment Details

B.1 OOD Detection Evaluation Metrics

Conventionally [17, 16, 27], OOD detection is treated as a binary classification problem in which the trained detector is expected to assign a positive label for an OOD task if its estimated OOD score exceeds some threshold τ To evaluate the performance of the OOD detector, we use three metrics: area under the receiver operating characteristic curve (**AUROC**), area under the precision-recall curve (**AUPR**), and the false positive rate at N% true positive rate (**FPRN**), where N=95 in our experiments.

As discussed in [17], the **AUROC** and **AUPR** are holistic metrics that summarize the performance of a detection method across multiple thresholds. The AUROC can be thought of as the probability that an OOD example is given a higher OOD score than an ID example. Thus, a higher AUROC is better, and an uninformative detector has an AUROC of 50%. The AUPR is useful when OOD inputs are infrequent, as it takes the base rate of OOD inputs into account.

Whereas the previous two metrics represent the detection performance across various thresholds, the FPRN metric represents performance at one strict threshold. By observing performance at a strict threshold, we can make clear comparisons among strong detectors. The **FPRN** metric is the probability that an in-distribution example (ground-truth negative sample) raises a false alarm (detected as a positive sample) when N% of ground-truth OOD examples (positive samples) are correcty detected, so a lower FPRN is better. Capturing nearly all anomalies with few false alarms can be of high practical value.

B.2 Single and Multi-sinusoid Regression

Model Architecture Both the the data EBM $E_\omega(\mathbf{x}, \mathbf{y}, \phi_i)$ and the prior EBM are MLPs. Follow the encoder implementation in CNPs [8], $q_\psi(\phi_i | \mathcal{T}_i^s)$ composes of a within-task mean pooling operation sandwiched between two arbitrary learnable transformations parameterized by MLP, i.e., $q_\psi(\phi_i | \mathcal{T}_i^s) = \text{MLP}_{\psi_1}(\frac{1}{N_i^s} \sum_{j=1}^{N_i^s} \text{MLP}_{\psi_2}([\mathbf{x}_{ij}^s, y_{ij}^s]))$. The output of q_ψ is the task latent variable $\phi_i \in \mathbb{R}^2$.

Hyperparameters We use a training batch size of 50 and learning rate of 0.0005 for all methods. The additional method-specific hyperparameters are stated below

Metafun [53] num-inner-loop: 5; initial representation: zero; outer learning rate: 10^{-4}; initial inner learning rate: 0.1; Dropout rate: 0.0; Orthogonality penalty weight: 0.0; L2 penalty weight: 0.0.

MAML [6] batch size: 4; num-inner-loop: 5; inner learning rate: 0.01; outer learning rate: 0.001;

ABML [41] batch size: 4; num-inner-loop: 5; inner learning rate: 0.005; outer learning rate: 0.001; alpha 1.0; beta 0.01; num reparameterizatio samples: 4;

F-POACH-GP [43] prior outputscale: 2.0; prior lengthscale: 0.2; prior weight: 0.001; learnable prior mean: True; learnable prior covariance: True.

EBML-CNPs Prior SGLD η: 0.01; Data SGLD η: 0.001; num-SGLD-iter: 20; energy L2 penalty 1.0;

B.3 Drug Activity Prediction

Preprocessing Molecular Graph Inputs Each input $\mathbf{x}$ is a SMILES representation of a chemical molecule, which essentially is a list of string characters of variable length. To transform the SMILE representation into numerical values, we use a pre-trained SMILES-transformer [18] for converting the sring input $\mathbf{x}$ into vector representation $\tilde{\mathbf{x}}$ in $\mathbb{R}^{1024}$. We treat this $\tilde{\mathbf{x}}$ as the inputs for all methods.

Model Architecture We use the same EBML-CNPs architecture as in the sinusoids experiments. However, we expand the latent variable dimension to $\mathbb{R}^{128}$, and the number of neurons in each hidden layer to 256. Furthermore, additional Batch Normalization layers [20] are interleaved with layers of the MLPs.

Hyperparameters We use a training batch size of 10 and learning rate of 0.0005 for all methods. The additional method-specific hyperparameters are stated below

Metafun [53] num-inner-loop: 5; initial representation: zero; outer learning rate: 10^{-4}; initial inner learning rate: 0.1; Dropout rate: 0.0; Orthogonality penalty weight: 0.0; L2 penalty weight: 0.0.

MAML [6] batch size: 4; num-inner-loop: 5; inner learning rate: 0.001; outer learning rate: 0.001;

ABML [41] batch size: 4; num-inner-loop: 5; inner learning rate: 0.001; outer learning rate: 0.001; alpha 1.0; beta 0.01; num reparameterizatio samples: 4;

F-POACH-GP [43] prior outputscale: 1.0; prior lengthscale: 0.5; prior weight: 0.001; learnable prior mean: True; learnable prior covariance: True.

EBML-CNPs Prior SGLD η: 0.1; Data SGLD η: 0.1; num-SGLD-iter: 40; energy L2 penalty 0.1;

B.4 Meta-dataset Few-shot Classification

B.4.1 Details of EBML-SimpleCNAPs and EBML-TSA

Model Architecture TSA [28] pre-trains a feature representation using available ID training domains, and incooperates additional task-specific adaptation modules at test time in the form of residual-connected transformation matrices to each convolution block. The parameters of these modules are inferred by gradient descent on the support set from scratch at meta-testing. The transformed feature representations of the support set samples are then used to build the class prototypes in a non-parametric classifier for inference of the query sample labels. SimpleCNAPs [1] also first pre-trains a feature extractor on a large dataset, i.e., ImageNet. However, unlike TSA, SimpleCNAPs meta-learns task-specific adaptations *during meta-training* by learning a parametrized task-encoding function that estimates the task-specific modules in the form of additional FILM parameters from the support set . Similar to TSA, the adapted support set features are used to construct the class centers in a non-parametric predictive function for classification of the query samples.

Thus for both methods, the set of prototypes in each ID training task resemble a task-specific predictive function, and is a suitable choice as the meta-learned prior knowledge. By specifying the latent variable ϕ_i to be a set of class prototypes used in the cosine classifier of each ID meta-training task, our EBM prior $p_\lambda(\phi_i)$ resembles a distribution over task-specific predictive functions from the ID domains. The architecture for the Prior EBM is depicted in Figure 7.

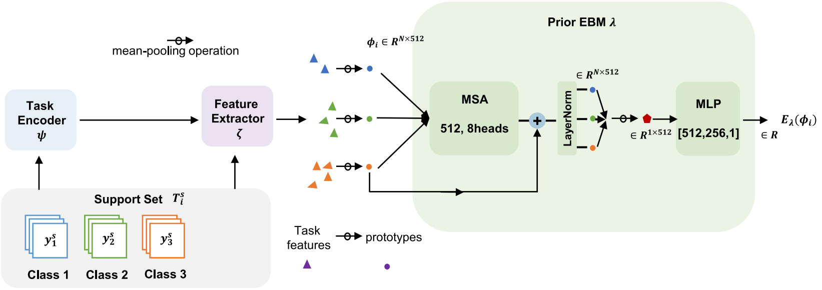


Figure 7: Model architecture for the prior EBM for classification.

To simultaneously achieve the best of both classification and OOD detection, we follow similar strategies in [50, 37] which learn an another decoder for prediction. The modified EBML architecture is shown in Figure 8.

Hyper-parameters We use the reported hyperparameters in TSA [28] and SimpleCNAPs [1] for training the base models. We report here the additional hyperparameters specific to EBML below in Table 6, which are found on the validation split of Meta-dataset [49].

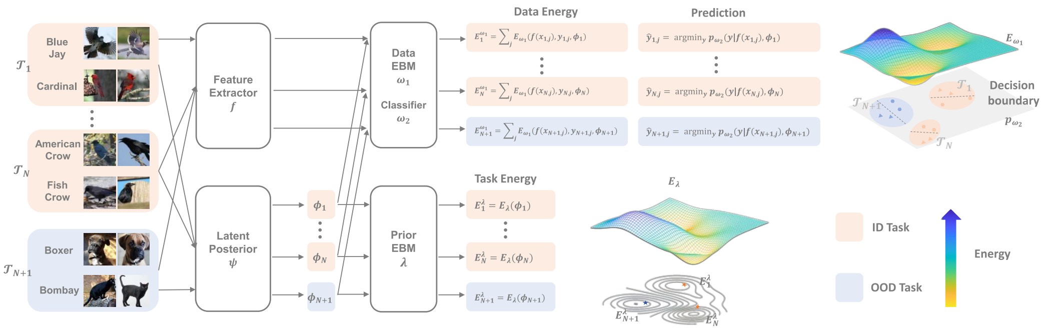

Data Energy
Prediction
Blue Jay
Cardinal
$\mathcal{T}_1$
American Crow
Fish Crow
$\mathcal{T}_N$
Boxer
Bombay
$\mathcal{T}_{N+1}$
Feature Extractor f
Latent Posterior ψ
Data EBM ω_1
Classifier ω_2
Prior EBM λ
ϕ_1
ϕ_N
ϕ_{N+1}
$E_1^{\omega_1} = \sum_j E_{\omega_1}(f(x_{1,j}), y_{1,j}, \phi_1)$
$E_N^{\omega_1} = \sum_j E_{\omega_1}(f(x_{N,j}), y_{N,j}, \phi_N)$
$E_{N+1}^{\omega_1} = \sum_j E_{\omega_1}(f(x_{N+1,j}), y_{N+1,j}, \phi_{N+1})$
$\hat{y}_{1,j} = \text{argmin}_y\, p_{\omega_2}(y|f(x_{1,j}), \phi_1)$
$\hat{y}_{N,j} = \text{argmin}_y\, p_{\omega_2}(y|f(x_{N,j}), \phi_N)$
$\hat{y}_{N+1,j} = \text{argmin}_y\, p_{\omega_2}(y|f(x_{N+1,j}), \phi_{N+1})$
E_{ω_1}
Decision boundary
p_{ω_2}
$\mathcal{T}_1$
$\mathcal{T}_N$
$\mathcal{T}_{N+1}$
Task Energy
$E_1^\lambda = E_\lambda(\phi_1)$
$E_N^\lambda = E_\lambda(\phi_N)$
$E_{N+1}^\lambda = E_\lambda(\phi_{N+1})$
E_λ
E_1^λ
E_N^λ
E_{N+1}^λ
ID Task
OOD Task
Energy

Figure 8: Overview of the EBML framework for image classification tasks. The task latent variable ϕ_i are inferred from the support set $\mathcal{T}_i^s$ following the implementation of the base algorithm. The data and task energy scores are evaluated by the data and prior EBMs E_{ω_1} and E_λ, respectively; while the query labels are predicted by the classifier p_{ω_2} of the base algorithm. The feature extractor f is a pre-trained ResNet-18 identical to the one used in SimpleCNAPs [1] and TSA [28].

Table 6: Hyperparameters for EBML on Meta-dataset 5-way 1-shot classification tasks.

Groups	Hyperparameters	Values
Training	num SGLD steps	40
	Data SGLD η	1.0
	Prior SGLD η	10.0
	Energy L2 penalty	1.0
	EBM Spectral Norm	True
Adaptation	num steps	10
	TSA learning rate β [28]	0.00091
	TSA learning rate α [28]	0.000267
	weight on prior energy	0.1
	m in prior energy margin loss	0.4
	optimizer	Adam

B.4.2 Baseline OOD Detection Methods for Classification

We used the following traditional OOD input detection methods as baselines for computing the task OOD scores in Table 2 main body of the paper. We compute the task OOD score for a task $\mathcal{T}_i$. as the average instance-level OOD scores over its input images $\{\mathbf{x}\}^{N_i}$. More specifically:

max softmax score [16], is taken as the maximum softmax prediction probability over N possible classes, that is $S_{\hat{y}}(\mathbf{x}) = \max_c \frac{\exp(logit_{[c]}(\mathbf{x}))}{\sum_{c=0}^{N-1} \exp(logit_{[c]}(\mathbf{x}))}$.

ODIN [30], extends the softmax-score by introducing temperature scaling and input preprocessing. More concretely, the input is perturbed following $\hat{\mathbf{x}} = \mathbf{x} + \epsilon\, sign(\nabla_{\mathbf{x}} \log S_{\hat{y}}(\mathbf{x}; T))$, which moves $\mathbf{x}$ in the direction that increases the temperature-scaled softmax score $S_{\hat{y}}(\mathbf{x}; T))$, computed as $\max_c \frac{\exp(logit_{[c]}(\mathbf{x})/T)}{\sum_{c=0}^{N-1} \exp(logit_{[c]}(\mathbf{x})/T)}$. The final ODIN confidence score writes $S_{\hat{y}}(\hat{\mathbf{x}}; T))$.

max logits score [15]. This is simply the maximal prediction logit of input image x, which is an alternative to the max-softmax prediction score.

MAH Detector [27]. We compute the MAH OOD score of $\mathbf{x}$ as the Mahalanobis distance from its feature representation to its nearest class mean which we estimate for each class using the empirical average of training inputs in class c. The shared covariance matrix used in estimating the Mahalanobis distance is computed on a subset of training samples from all training classes.

Domain Selector. When given the domain-IDs during training, a few methods [34] on Meta-dataset classification problems adapt a domain classifier network for inferring the task-specific parameters from a set of candidate domain-specific feature modulation parameters. The max softmax score of the trained domain selector can be used to compute the OOD score for $\mathbf{x}$.

The perturbation magnitude ϵ and the temperature scale T used in ODIN and MAH are determined using the validation set of meta-dataset, with a a grid-search over the parameter space for $\epsilon \in \{0, 0.00002, 0.00005, 0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005\}$ and $T \in \{1, 2, 10, 50, 100, 200, 500\}$. ODID and MAH on average improves the OOD task detection results using max softmax and max logits scores for the same baseline model.

C Additional Experiment Results

C.1 Multi-sinusoids Few-shot Regression and OOD Task Detection

Task generation The multi-target regression tasks are synthesised by superposing each generated sinusoid from the ID and OOD distribution with a phase-shifted version of itself at a constant phase lag of 0.3π, such that now in each task every input x has two possible target values y. We give both y values for each x in the support set.

Results In Table 7, 8 we see that EBML-CNPs achieved both the best regression and OOD detection performance. In Table 9, our proposed energy outperform all ablated models.

Table 7: Regression performance for multi-sinusoids experiments.

Models	ID NLL↓
ABML [41]	0.886±0.048
F-PACOH-GP [43]	1.289±0.023
CNPs [8]	0.865±0.069
Metafun[53]	0.874±0.051
EBML-CNPs	**0.282** ±0.041

Table 8: OOD detection performance on multi-sinusoids tasks.

OOD Scores	Models	AUROC↑	AUPR↑	FPR95↓
Std	ABML [41]	74.14	72.15	73.67
	Metafun [53]	76.52	77.34	88.12
SNLL	ABML [41]	54.75	56.32	99.60
	F-PACOH-GP [43]	55.24	68.95	100.00
	CNPs [8]	70.43,	79.71	92.4
	Metafun [53]	79.21	77.03	90.98
	EBML-CNPs (Ours)	92.77	94.25	46.20
Energy Sum	EBML-CNPs (Ours)	**94.91**	**96.15**	**34.60**

Table 9: Ablation study for Energy Sum on Multi-sinusoids few-shot regression tasks.

OOD Scores	Models	AUROC↑	AUPR↑	FPR95↓
ABML [41]	SNLL	54.75	56.32	99.60
	+Gauss Prior	86.64	86.45	50.00
CNPs [8]	SNLL	70.19	79.49	95.20
	+Gauss Prior	82.90	87.23	76.60
EBML-CNPs	SNLL	92.77	94.25	46.20
	+EBM Prior	**94.91**	**96.15**	**34.60**

C.2 Meta-dataset 5-way-1-shot Classification and OOD Task Detection

C.2.1 TSA-EBML vs TSA on Unshuffled 5-way-1-shot Meta-dataset Tasks

To ensure our few-shot classification results in Table 5 are fair and up-to-date, we follow the latest evaluation protocols in Meta-dataset which sets *shuffle_buffer_size*=1000, and test TSA (reproduced using their official code) and EBML-TSA on the same set of sampled testing tasks. However, the official 5-way-1-shot classification results of TSA (Table 8 in [28]) are reported on an earlier version of Meta-dataset before the fix of issue 54 [2]. This explains the observed differences between TSA results in Table 5 and Table 8 in [28].

In this section, we verify that the improved classification performance of EBML-TSA over TSA is indeed **not** a result of the change in evaluation protocols in Meta-dataset. To do so, we evaluate EBML-TSA under identical settings to TSA in Table 8 of [28], i.e., 5-way-1-shot settings with *shuffle_buffer_size*=0.

In Table 10, we compare our results with the official results of TSA. The performance of both methods are largely similar to that in Table 5, except that the classification accuracy for

Table 10: Classification performance on Meta-dataset 5-way 1-shot tasks, with *shuffle_buffer_size*=0. * indicates results reported by [28].

Datasets	**TSA*** [28]	**EBML-TSA** (Ours)
Omniglot	**96.3**±0.4	**96.3**±0.5
Textures	**54.5**±0.9	**54.5**±0.8
Aircraft	**79.6**±0.9	79.0±0.9
Birds	74.5±0.9	**75.3**±0.9
VGG Flower	**80.3**±0.8	80.2±0.8
Fungi	75.3±1.0	**77.1**±0.9
Quickdraw	79.3±0.9	**79.9**±0.9
MSCOCO	59.9±1.0	**60.2**±1.0
Traffic Sign	57.2±1.0	**58.2**±0.9
CIFAR10	55.8±0.9	**56.8**±0.9
CIFAR100	63.7±1.0	**64.6**±1.0
MNIST	80.1±0.9	**82.0**±0.9
Avg ID	77.1	**77.5**
Avg OOD	63.4	**64.4**
Avg All	71.4	**72.0**

---

[2] As mentioned in https://github.com/google-research/meta-dataset/issues/54, the *shuffle_buffer_size* was set to zero in an earlier version of Meta-dataset which can lead to some biased results in evaluation.

Table 11: 5-way-1-shot classification accuracy of OOD tasks in Meta-datasets [49].

Datasets	**SimpleCNAPs** [1]	**EBML-SimpleCNAPs** (Ours)	**URL** [29]	**EBML-URL** (Ours)
MSCOCO	49.37±0.99	51.75±0.96	59.14±0.95	59.82±0.97
Traffic Sign	55.63±0.96	56.12±0.97	57.62±0.90	58.26±0.90
CIFAR10	50.79±0.85	51.16±0.89	54.37±0.83	54.39±0.86
CIFAR100	54.15±0.95	55.23±0.93	63.03±0.97	62.74±0.96
MNIST	80.25±0.85	81.01±0.85	78.85±0.86	79.78±0.87

both methods improved on a few datasets e.g., MNIST, MSCOCO. Noticeably, EBML-TSA still outperforms TSA, on 5/7 ID domains (2/5 equal performance) with an average increase of 0.4% in accuracy, and 5/5 OOD domains with an average improvement of 1.0%. The results validate the effectiveness of our proposed method and verify that our methods indeed is not favoured by the latest evaluation protocol in Meta-dataset.

C.2.2 5-way-1-shot OOD classification results in Meta-dataset [49] for more EBML variants

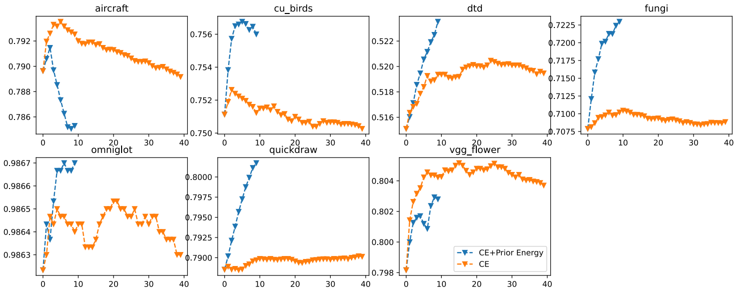
We instantiate EBML with two additional baseline Meta-learning algorithms including SimpleCNAPs [1] and URL [29] and report their 5-way-1-shot classification accuracy of OOD tasks in Meta-datasets [49] in Table 11 above. OOD-adaptation for EBML is performed by optimizing Eqn. (11) w.r.t. to the task encoder that produces the task-specific FiLM in SimpleCNPAS, and w.r.t. the feature projection matrix in URL.

C.2.3 Classification Accuracy Trajectories During OOD Task Adaptation

In Figure 9 and 10, we visualize the average query set classification accuracy throughout the OOD task adaptation for TSA and EBML-TSA. Results for TSA are produced using the official optimal hyperparameters reported in [28]; while for EBML-TSA, we use the hyperparameters reported in Table 6.

The y-axis in plot represents the average classification accuracy on the query set, while the x-axis represents the steps during OOD adaptation. We observe that our objective (Blue) in Eqn. (11) generally alleviates the over-fitting behaviour in the adaptation process caused by minimizing the cross-entropy loss alone in TSA (Orange). This meta-regularization effect is more apparent on tasks from the OOD domains.

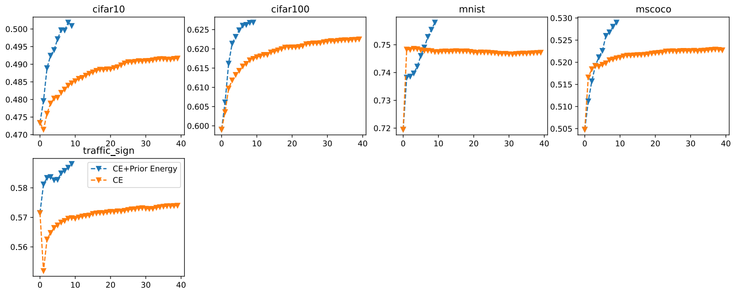
Figure 9: TSA vs EBML-TSA query set classification accuracy during adaptation on ID datasets



C.2.4 More OOD Adaptation Baselines on Meta-dataset

In the experiment on Meta-dataset 5-way 1-shot classification tasks, we assigned pseudo-labels to the query inputs and used them together with the labelled support set samples in calculation of the class prototypes, hence the prior energy score for each task, in Eqn. (11). Therefore, In this study, we compare our results with several baselines that also utilize the unlabelled query set in for adaptation, including:

Figure 10: TSA vs EBML-TSA query set classification accuracy during adaptation on OOD datasets



Query Entropy (TSA+QE), which minimizes the average entropy of the prediction distribution on the query samples in addition to the support set classification loss.
Confidently-predicted Pseudo-labels (TSA+PL), which tunes a confidence threshold, assigns query predictions over the threshold as the pseudo-labels, and then minimizes the classification loss on the support set and these confidently-predicted query samples before testing.

The optimal results are reported in Table 12. The results show that while the two baseline methods exploiting the query set information can achieve better performance than TSA on some datasets, they do not outperform EBML-TSA with using Eqn. (11) for task adaptation.

Table 12: Additional 5-way 1-shot classification results when using TSA with 1) entropy minimization on the query set, and 2) cross-entropy on confidently-predicted pseudo-labelled query samples, for OOD task adaptation.

Datasets	**EBML-TSA**	**TSA+EM**	**TSA+PL**
Omniglot	98.67±0.26	98.81±0.26	98.25±0.25
Textures	52.35±0.88	52.12±0.87	51.91±0.87
Aircraft	78.47±0.86	79.40±0.86	79.09±0.86
Birds	75.52±0.90	74.76±0.89	75.07±0.90
VGG Flower	80.30±0.83	80.00±0.81	80.71±0.80
Fungi	72.29±0.94	70.95±0.93	70.44±0.94
Quickdraw	80.27±0.85	79.18±0.85	78.96±0.85
MSCOCO	53.03±0.97	52.24±0.94	52.55±0.94
Traffic Sign	58.85±1.01	57.13±0.95	57.06±0.94
CIFAR10	50.04±0.89	50.22±0.81	49.43±0.83
CIFAR100	62.77±1.05	62.47±1.00	62.70±0.99
MNIST	76.08±0.88	75.23±0.88	75.14±0.85
Avg ID	76.84	76.86	76.35
Avg OOD	60.15	59.46	59.38
Avg All	**69.89**	68.16	69.28

C.2.5 Baseline OOD Adaptation is Less Reliable without Sufficient Support Samples

We intend to use this experiment as an empirical evidence to support our argument on that *SOTA cross-domain meta-learning algorithms produce unreliable task-specific adaptation without sufficient support set samples*. We train Simple-CNAPs [1] and TSA [28] following the official experimental setup of Meta-dataset [49], except that we have excluded ImageNet in the ID training datasets due to limited computation resources.

Following [1, 28], in the varying-way varying-shot testing configuration, the number of classes in each task varies between 5 to 50, while the total number of support samples per task varies between 5 to 500. The maximal number of support samples per class is capped at 100. We report the average testing accuracy over 600 tasks for the varying-way varying shot and 5-way 1-shot settings in Table 13 below. We observe that the average classification accuracy for both ID and OOD domains generally decrease for 5-way 1-shot tasks, which suggests that the adaptation with without sufficient support set samples is inherently difficult.

Table 13: Classification accuracy of SimpleCNAPs and TSA on meta-dataset for varying-way varying-shot vs 5-way 1-shot meta-testing configurations.

	Varying-Way Varying-Shot		**5-Way 1-Shot**	
Datasets	**SimpleCNAPs** [1]	**TSA** [28]	**Simple-CNAPs** [1]	**TSA** [28]
Omniglot	91.61±0.57	94.77±0.41	97.87±0.27	98.63±0.26
Textures	65.47±0.71	77.06±0.67	44.11±0.83	51.93±0.87
Aircraft	81.28±0.69	88.56±0.51	65.08±0.89	78.91±0.86
Birds	73.80±0.81	80.86±0.77	66.21±0.98	75.02±0.90
VGG Flower	90.15±0.50	92.48±0.52	76.57±0.83	80.37±0.80
Fungi	44.82±1.13	66.49±1.02	50.95±0.95	70.89±0.93
Quickdraw	73.30±0.81	82.33±0.58	67.61±0.95	79.02±0.84
MSCOCO	35.19±0.94	55.22±1.09	38.78±0.79	52.28±0.94
Traffic Sign	42.59±1.01	82.60±0.97	50.84±0.90	57.40±0.94
CIFAR10	56.65±0.80	80.40±0.71	37.59±0.67	49.16±0.82
CIFAR100	44.13±1.10	70.38±0.97	46.11±0.89	62.25±1.01
MNIST	93.89±0.37	96.44±0.43	76.52±0.83	74.72±0.83
Avg ID	**74.35**	**83.22**	66.91	76.40
Avg OOD	**54.49**	**77.01**	49.97	59.16
Avg All	**66.07**	**80.63**	59.85	69.22

C.3 EBML with Probabilistic Posterior Distribution $q_\psi(\phi_i | \mathcal{T}_i^s)$

Since EBML is a flexible plug-in, it is definitely compatible with those methods that use probabilistic posterior q_ψ. However, we currently mainly focus on the MAP estimate in the main paper because the majority of the state-of-the-art algorithms [8, 28] that EMBL has applied to resort to a MAP estimate for q_ψ.

Nevertheless, in the Table 14 below, we show the results of EBML with Neural Processes (NPs) [9], named EBML-NPs, on sine regression tasks. NPs parameterizes q_ψ as a multivariate Gaussian distribution whose task-specific mean and standard deviation are determined by a learnable neural network encoder conditioned on the task support set.

For training EBML-NPs, we include the the entropy term $\mathcal{H}(q_\psi(\phi_i | \mathcal{T}_i^s))$ in the training objective in Eqn. (8), which has a closed-form solution when q_ψ is a Gaussian. We resort to the re-parameterization trick for computing the expectations $\mathbb{E}_{q_\psi}$ in Eqn. (8). When using energy sum for OOD detection, we also add the entropy term $\mathcal{H}(q_\psi(\phi_i | \mathcal{T}_i^s))$ for EBML-NPs for consistency.

Table 14: Regression and OOD detection performance on sinusoids few-shot regression tasks for EBML-NPs vs EBML-CNPs.

OOD Scores	**EBML-NPs**				**EBML-CNPs**			
	MSE	**AUROC**	**AUPR**	**FPR95**	**MSE**	**AUROC**	**AUPR**	**FPR95**
SNLL	0.009±0.002	95.94	97.16	31.20	0.009±0.002	96.46	97.41	29.40
Energy Sum w/o Entropy		97.10	97.83	20.40		97.74	98.31	14.20
Energy Sum w Entropy		97.58	97.79	14.80		n/a	n/a	n/a

In Table 14, we conclude that EBML-NPs with probabilistic q_ψ achieves comparable performance to its deterministic version, EBML-CNPs; there is no significant improvement in performance by switching to a probabilistic q_ψ. Nevertheless, we observe that both the prior energy function and the extra entropy terms bring in positive contribution to the OOD detection performance compared to SNLL alone.

C.4 Computational Complexity Analysis

We conduct a computational complexity analysis for EBML by comparing its wall-clock training time and convergence to baselines, results are shown above in Table 15 and Figure 11.

Table 15: Training time in seconds. $*$ ABML is much slower due to the gradient-based inner loop optimizations and learning with a Bayesian Neural Network, which makes it challenging to parallelize training over a batch of tasks.

Sinusoids	CNPs [8]	EBML-CNPs	f-PACOH-GP [43]	ABML [41]
Training time / 500 tasks	0.67	1.83	3.13	11.95*

Meta-dataset	SimpleCNAPs [1]	EBML-SimpleCNAPs
Training time / 1000 tasks	1.02	1.75

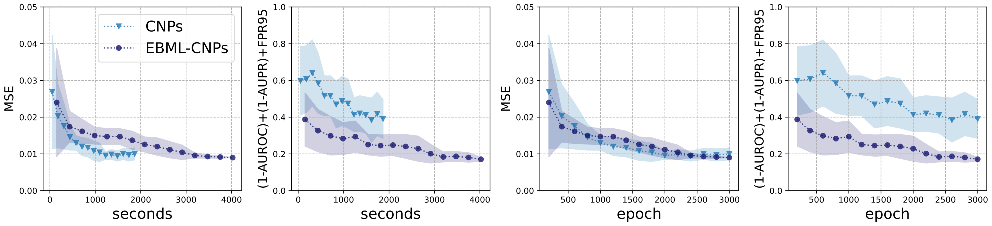


Figure 11: **Left** : Wall-clock convergence in seconds, and **Right**: performance vs number of training epochs, for EBML-CNPs vs CNPs in single-sinosoid few-shot regression tasks. The plots show the regression (MSE ↓) and combined OOD tasks detection (1-AUROC)+(1-AUPR)+FPR95 ↓ performance on single sine few-shot regression tasks during training. Curves are moving averages with window size 3. EBML-CNPs achieves better final performance than CNPs.

From the results above, EBML is computationally cheaper and faster than the two Bayesian methods, namely, F-PACOH-GP [43] which requires matrix inversion for inference with Gaussian processes prior, and ABML [41] which imposes a Gaussian prior over the entire parameter space of the model. Meanwhile, in Table 4 and 1 in the paper, EBML achieves the best regression and OOD detection performance out of all baselines.

D Empirical Study on Distribution Shift in the Input Space

In real-word applications, distribution shift in input space, i.e. the distribution shift in $\mathbf{X}$, is a very common phenomenon. Take AI-aided drug discovery as an example, when predicting the bio-activities of a molecule for a given target protein, we may encounter molecules with very different molecular sizes, scaffolds etc, from the training examples [21]. Such input distribution shift, unfortunately, cannot always be correctly reflected by models trained to maximize the predictive probability $p_\omega(y|\mathbf{x}, \phi_i)$.

We first conduct a controlled experiment and show that modelling the joint distribution can lead to superior performance in OOD task detection which further substantiate our claim. We base our experiment on drug activity prediction tasks as described in Section 5, where $p(x)$ changes across tasks. The experimental details are as follow.

Setup There are three major factors affecting OOD detection performance, including a) whether we model the conditional or joint distribution, b) the model capacity, e.g., Gaussians or EBMs, and c) OOD scores, e.g., energy sum or sum of negative log-likelihood (SNLL) of the support samples. To investigate the effect of (a) specifically, we fix the controlled variables (b) with the same EBM architectural capacity, and (c) with either energy sum or SNLL. Consequently, we compare **EBML-joint**, which is exactly our proposed training procedure in the paper, and **EBML-conditional**, which follows the same training with EBML-joint but models $p(\mathbf{Y}|\mathbf{X})$ instead of $p(\mathbf{X}, \mathbf{Y})$ of the meta-training task distribution. Concretely, the training objective for EBML-conditional becomes

$$\begin{aligned} &\arg\max_{\omega,\lambda,\psi} \log p_{\omega,\lambda,\psi}(Y_i|X_i) := \\ &\text{argmax}_{\omega,\lambda,\psi}\, \mathbb{E}_{\phi_i\sim q_\psi(\phi_i|\mathcal{T}_i^s)} \Big[\sum_j -E_\omega(y_{ij}^s, x_{ij}^s, \phi_i) + \mathbb{E}_{y'\sim p_\omega(y'|x_{ij}^s,\phi_i)}[E_\omega(y'_{ij}, x_{ij}^s, \phi_i)]\Big] \\ &- \mathbb{E}_{\phi_i\sim q_\psi(\phi_i|\mathcal{T}_i^s)}[-E_\lambda(\phi_i)] + \mathbb{E}_{\phi_i\sim p_\lambda(\phi_i')}[E_\lambda(\phi_i')] + \mathcal{H}(q_\psi(\phi_i|\mathcal{T}_i^s)), \end{aligned} \tag{24}$$

which only differs from the training objective of EBML-joint in Eqn. (8) in the sampling $y' \sim p_\omega(y'|x_{ij}^s, \phi_i)$.

Table 16: EBML-joint vs EBML-conditional on DrugOOD few-shot regression and OOD task detection.

OOD Scores	EBML-joint				EBML-Conditional			
	Mean R^2	AUROC	AUPR	FPR95	Mean R^2	AUROC	AUPR	FPR95
SNLL	0.533	99.71	99.71	2.20	0.534	54.91	54.14	66.60
Energy Sum		99.79	99.78	1.40		63.74	58.15	66.20

Results In Table 16, we observe that EBML-joint outperforms EBML-conditional by large margins in detecting OOD tasks (molecules with larger molecular size).

For an additional illustration, in Figure 12, we show the histogram of the averaged support samples negative log-likelihood (SNLL) of a CNPs model trained with $p(\mathbf{Y}|\mathbf{X})$. We see that CNPs still outputs relatively high likelihood for some of the OOD tasks making their prediction indistinguishable from the ones on ID tasks

These empirical evidence support our motivation for modelling the joint distribution instead of the conditional distribution for potentially achieving better OOD task detection performance.

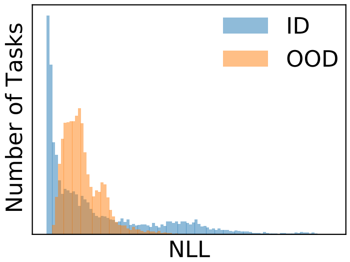


Figure 12: Histogram for SNLL of CNPs trained with $p(\mathbf{Y}|\mathbf{X})$ of ID and OOD tasks, where OOD tasks contain molecules with much larger molecular sizes than the ones in ID tasks.

E Pseudocode for EBML

Algorithm 1 EBML Meta-training

```
Input: Meta-training tasks $\{\mathcal{T}_1, \mathcal{T}_2, ... \mathcal{T}_N\}$
Output: Meta-learned optimal parameters $\omega*$, $\psi*$, $\lambda*$
Initialize the parameters $\omega$,$\psi$,$\lambda$ for data EBM, latent posterior, and prior EBM
while not converged do
    $B \sim \{\mathcal{T}_1, \mathcal{T}_2, ... \mathcal{T}_N\}$    ▷ sample a batch of tasks
    for $i = 1, 2, ..., |B|$ do    ▷ for each sampled task in $\mathcal{B}$
        $\mathcal{T}_i^s = \{\mathbf{X}_i^s, \mathbf{Y}_i^s\}^{N_i^s}, \mathcal{T}_i^q = \{\mathbf{X}_i^q, \mathbf{Y}_i^q\}^{N_i^q} \sim \mathcal{T}_i$    ▷ sample support and query sets
        $\phi_i \sim q_\psi(\phi_i | \mathcal{T}_i^s)$    ▷ infer the task latent variable by the base algorithm
        $x', y' \sim p_\omega(x', y' | \phi_i)$, $\phi_i' \sim p_\lambda(\phi_i')$    ▷ Sampling by SGLD in Eqn. (3)
        Compute loss for $\mathcal{T}_i$ as $\mathcal{L}_i(\omega, \psi, \lambda)$ using Eqn. (8).
    end for
    $\omega, \psi, \lambda \leftarrow$ Opt($\nabla_{\omega,\psi,\lambda} \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \mathcal{L}_i$)    ▷ Update parameters in the outer-loop
end while
```

Algorithm 2 EBML Meta-testing

```
Input: Meta-testing tasks $\{\mathcal{T}_1, \mathcal{T}_2, ... \mathcal{T}_N\}$, parameters $\lambda^*, \omega^*, \psi^*$, num adaptation steps $K$
Output: Query Prediction for each task $\{\mathbf{Y}_1^q, \mathbf{Y}_2^q, ...\mathbf{Y}_N^q\}$
for $i = 1, 2, ..., N$ do    ▷ for each test task
    $\mathcal{T}_i \rightarrow \mathcal{T}_i^s = \{\mathbf{X}_i^s, \mathbf{Y}_i^s\}^{N_i^s}, \mathcal{T}_i^q = \{\mathbf{X}_i^q, \}^{N_i^q}$    ▷ get support and unlabelled query sets
    if $K > 0$ then    ▷ Using EBML OOD task adaptation
        $\zeta \leftarrow$ Alg. (3)($\lambda*, \omega*, \psi*, \mathcal{T}_i^s, K$)    ▷ EBML OOD Task Adaptation
    else
        $\zeta \leftarrow \emptyset$
    end if
    $y_j^q = \arg\min_y \mathbb{E}_{\boldsymbol{\phi} \sim q_{\psi* \cup \zeta}(\boldsymbol{\phi} | \mathcal{T}_i^s)} \left[ E_{\omega*}(\mathbf{x}_j^q, y, \boldsymbol{\phi}) + E_{\lambda*}(\boldsymbol{\phi}) \right], \forall j \in \mathbf{X}_i^q$    ▷ query prediction
end for
```

Algorithm 3 EBML OOD Task Adaptation

```
Input: Model parameters $\lambda^*, \omega^*, \psi^*$, task support set $\mathcal{T}_i^s$, num adaptation steps $K$
Output: Task-specific parameter $\zeta$
for $k = 1, 2, ...K$ do    ▷ for $K$ adaptation steps
    $\phi_i \sim q_{\psi* \cup \zeta}(\phi_i | \mathcal{T}_i^s)$    ▷ infer task latent variable by base algorithm
    Compute loss on $\mathcal{T}_i^s$ as $\mathcal{L}(\zeta)$ using Eqn. (11).
    $\zeta \leftarrow$ Opt($\nabla_\zeta \mathcal{L}(\zeta)$)    ▷ Update task-specific parameter $\zeta$
end for
```