SUPPLEMENTARY MATERIAL

A. Regression experiments

A.1. Training configuration

As mentioned in section 4.1, the experiment is carried out on a multi-modal structured data, where a half of tasks are generated from sinusoidal functions, while the other half of tasks are from linear functions. The sinusoidal functions are in the form of $A\sin(x+\varphi)$, where the amplitude A and the phase φ are uniformly sampled from [0.1, 5] and [0, π], respectively. The linear functions are in the form of $ax+b$, where the slope a and the intercept b are sampled from the uniform distribution on [-3, 3]. The input x is uniformly sampled from [-5, 5]. In addition, a Gaussian noise with zero-ed mean and a standard deviation of 0.3 is added to the output.

The model used in this experiment is a 3-hidden fully connected neural network with 100 hidden units per each hidden layer. Output from each layer is activated by ReLU without batch normalisation. The optimisation for the objective function in (3) is carried out by Adam. Note that for regression, there is we do not place any weighting factor for the KL divergence term of VFE. Please refer to Table 3 for the details of hyperparameters used.

Hyperparameters	**Notation**	**Value**
Learning rate for variational parameters	α	0.001
Number of gradient updates for variational parameters		5
Number of Monte Carlo samples for variational parameters	L_t	128
Number of tasks before updating meta-parameters	T	10
Learning rate for meta-parameters	γ	0.001
Number of Monte Carlo samples for meta-parameters	L_v	128

Table 3: Hyperparameters used in the regression experiments on multi-modal structured data.

A.2. Additional results

In addition to the results in Figure 2, we also provide more qualitative visualisation from the multi-modal task distribution in Figure 4.

We also implement many Bayesian meta-learning methods, such as PLATIPUS, BMAML and Amortised Meta-learner, to compare with VAMPIRE using reliability diagram. We train all the methods of interest in the same setting used for VAMPIRE to obtain a fair comparison. The mean-squared error (MSE) of each method after training can be referred to Table 4. Please note that for probabilistic methods, MSE is the average value across many Monte Carlo samples or particles sampled from the posterior distribution of model parameters.

Method	**MSE**
MAML	1.96
PLATIPUS	1.86
BMAML	1.12
Amortised Meta-learner	2.32
VAMPIRE	2.24

Table 4: Mean squared error of many meta-learning methods after being trained in the same setting are tested on 1000 tasks.

B. Classification experiments

This section describes the detailed setup to train and validate the few-shot learning on Omniglot and mini-ImageNet presented in Sec. 4.2. Following the notation used in Sec. 3.1, each task or episode i has N classes, where the support set $\mathcal{Y}_i^{(t)}$ has k samples per class, and the query set $\mathcal{Y}_i^{(v)}$ has 15 samples per class. This is to be consistent with the previous works in the literature [16, 19]. The training is carried out by using Adam to minimise the cross-entropy loss of the softmax output.

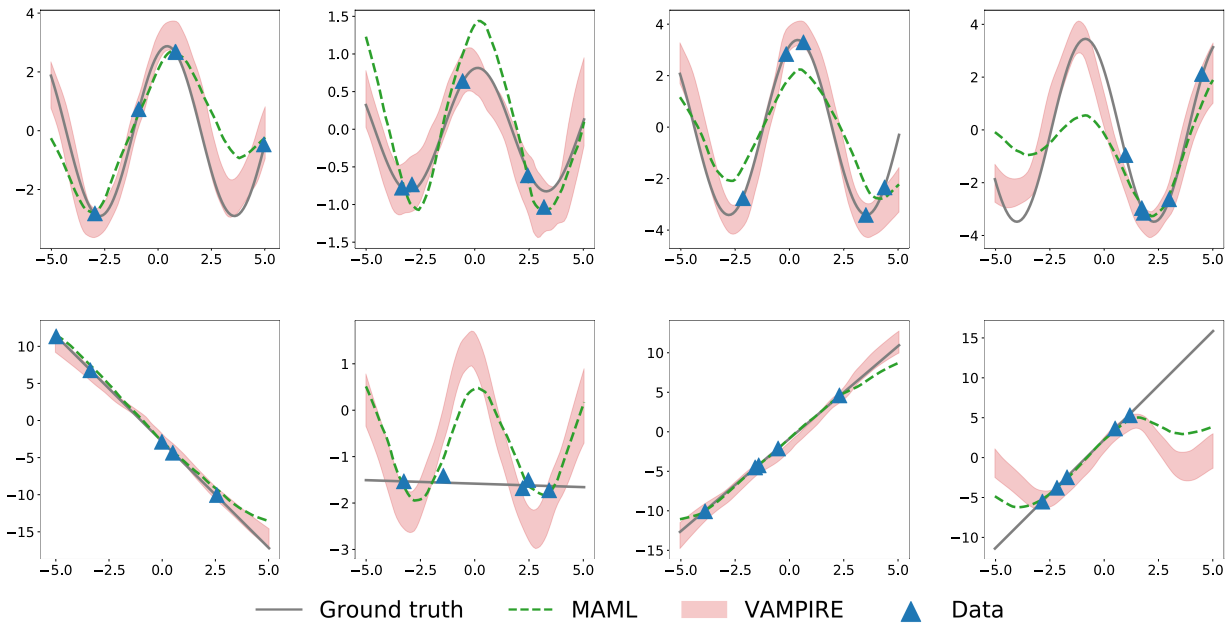

Figure 4: Additional qualitative results with tasks generated from either sinusoidal or linear function. The shaded area is the prediction made by VAMPIRE $\pm$ 1$\times$ standard deviation.

The learning rate of the meta-parameters θ is set to be $\gamma = 10^{-3}$ across all trainings, and decayed by a factor of 0.99 after every 10,000 tasks. Other hyperparameters used are specified in Table 5. We select the number of ensemble models L_t and L_v to fit into the memory of one Nvidia 1080 Ti GPU. Higher values of L_t and L_v are desirable to achieve a better Monte Carlo approximation.

Description	Notation	Omniglot		mini-ImageNet
		5-way	20-way	5-way
Number tasks per meta-update	T	32	16	2
Number of ensemble models (train)	L_t (train)	1	1	10
Number of ensemble models (train)	L_v (train)	1	1	10
Number of ensemble models (test)	L_t (test)	10	10	10
Number of ensemble models (test)	L_v (test)	10	10	10
Learning rate for $\mathbf{w}_i$	α	0.1	0.1	0.01
Learning rate for θ	γ	10^{-3}	10^{-3}	10^{-3}
Number of inner gradient updates		5	5	5

Table 5: Hyperparameters used in the few-shot classification presented in Sec. 4.

For the experiments using extracted features [22] presented in Table 6 for mini-ImageNet, and the bottom part of Table 2 for tiered-ImageNet, we used a 2-hidden fully connected layer with 128 and 32 hidden units. The learning rate α is set as 0.01 and 5 gradient updates were carried out. The learning rate for meta-parameters was $\gamma = 0.001$.

Both the experiments for classification re-weight the KL divergence term of VFE by a factor of 0.1.

B.1. Model calibration for classification - ECE and MCE

We provide the results of model calibration, in particular, ECE and MCE in the numeric form. We also include the 95% confidence interval in Table 7, although they are extremely small due to the large number of unseen tasks.

	MINI-IMAGENET [16]	
	1-SHOT	5-SHOT
NON-STANDARD CNN		
RELATION NETS [41]	50.44 ± 0.82	65.32 ± 0.70
VERSA [29]	53.40 ± 1.82	67.37 ± 0.86
SNAIL [48]	55.71 ± 0.99	68.88 ± 0.92
ADARESNET [49]	56.88 ± 0.62	71.94 ± 0.57
TADAM [50]	58.5 ± 0.30	76.7 ± 0.30
LEO [22]	61.76 ± 0.08	77.59 ± 0.12
METAOPTNET [46]	$\mathbf{64.09 \pm 0.62}$	$\mathbf{80.00 \pm 0.45}$
VAMPIRE	62.16 ± 0.24	76.72 ± 0.37

Table 6: Accuracy for 5-way classification on mini-ImageNet tasks (in percentage) of many methods which uses extra parameters, deeper network architectures or different training settings.

Method	**ECE**	**MCE**
MAML	0.0410 ± 0.005	0.124
PLATIPUS	0.032 ± 0.005	0.108
BMAML	0.025 ± 0.006	0.092
Amortised Meta-learner	0.026 ± 0.003	0.058
VAMPIRE	0.008 ± 0.002	0.038

Table 7: Results of ECE and MCE of several meta-learning methods that are tested in 5-way 1-shot setting over 15504 unseen tasks sampled from mini-ImageNet dataset.

C. Pseudo-code for evaluation

Algorithm 2 VAMPIRE testing

Require: a new task $\mathcal{T}_{T+1}$, θ, L_t, L_v, α and β

1: $\lambda_{T+1} \leftarrow \boldsymbol{\theta}$

2: sample $\hat{\mathbf{w}}_{T+1}^{(l)} \sim q(\mathbf{w}_{T+1}|\lambda_{T+1})$, where $l_t = 1 : L_t$

3: update: $\lambda_i \leftarrow \lambda_i - \frac{\alpha}{L_t} \nabla_{\lambda_i} \mathcal{L}_i^{(t)}|_{\mathcal{Y}_{T+1}^{(t)}}$

4: draw L_v ensemble model parameters $\hat{\mathbf{w}}_i^{(l_v)} \sim q(\mathbf{w}_i; \lambda_i)$

5: compute prediction $\hat{\mathcal{Y}}_i^{(v)}$ using L_v ensemble models.