Supplementary Materials: MetaKernel: Learning Variational Random Features with Limited Labels

Yingjun Du, Haoliang Sun, Xiantong Zhen, Jun Xu, Yilong Yin, Ling Shao, *Fellow, IEEE* and Cees G. M. Snoek, *Senior Member, IEEE*

1 Derivations of the ELBO

For a singe task, we begin with maximizing log-likelihood of the conditional distribution $p(\mathbf{y}|\mathbf{x}, \mathcal{S})$ to derive the ELBO of MetaKernel. By leveraging Jensen's inequality, we have the following steps as

$$\begin{aligned}
\log p(\mathbf{y}|\mathbf{x}, \mathcal{S}) &= \log \int p(\mathbf{y}|\mathbf{x}, \mathcal{S}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathbf{x}, \mathcal{S}) d\boldsymbol{\omega} && (1)\\
&= \log \int p(\mathbf{y}|\mathbf{x}, \mathcal{S}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathbf{x}, \mathcal{S}) \frac{q_\phi(\boldsymbol{\omega}|\mathcal{S})}{q_\phi(\boldsymbol{\omega}|\mathcal{S})} d\boldsymbol{\omega} && (2)\\
&\geq \int \log \left[\frac{p(\mathbf{y}|\mathbf{x}, \mathcal{S}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathbf{x}, \mathcal{S})}{q_\phi(\boldsymbol{\omega}|\mathcal{S})} \right] q_\phi(\boldsymbol{\omega}|\mathcal{S}) d\boldsymbol{\omega} && (3)\\
&= \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{\omega}|\mathcal{S})} \log \left[p(\mathbf{y}|\mathbf{x}, \mathcal{S}, \boldsymbol{\omega}) \right] - D_{\mathrm{KL}}[q_\phi(\boldsymbol{\omega}|\mathcal{S})||p(\boldsymbol{\omega}|\mathbf{x}, \mathcal{S})]}_{\text{ELBO}}. && (4)
\end{aligned}$$

The ELBO can also be derived from the perspective of the KL divergence between the variational posterior $q_\phi(\boldsymbol{\omega}|\mathcal{S})$ and the posterior $p(\boldsymbol{\omega}|\mathbf{y}, \mathbf{x}, \mathcal{S})$:

$$\begin{aligned}
D_{\mathrm{KL}}[q_\phi(\boldsymbol{\omega}|\mathcal{S})||p(\boldsymbol{\omega}|\mathbf{y}, \mathbf{x}, \mathcal{S})] &= \mathbb{E}_{q_\phi(\boldsymbol{\omega}|\mathcal{S})} \left[\log q_\phi(\boldsymbol{\omega}|\mathcal{S}) - \log p(\boldsymbol{\omega}|\mathbf{y}, \mathbf{x}, \mathcal{S}) \right]\\
&= \mathbb{E}_{q_\phi(\boldsymbol{\omega}|\mathcal{S})} \left[\log q_\phi(\boldsymbol{\omega}|\mathcal{S}) - \log \frac{p(\mathbf{y}|\boldsymbol{\omega}, \mathbf{x}, \mathcal{S}) p(\boldsymbol{\omega}|\mathbf{x}, \mathcal{S})}{p(\mathbf{y}|\mathbf{x}, \mathcal{S})} \right]\\
&= \log p(\mathbf{y}|\mathbf{x}, \mathcal{S}) + \mathbb{E}_{q_\phi(\boldsymbol{\omega}|\mathcal{S})} \left[\log q_\phi(\boldsymbol{\omega}|\mathcal{S}) - \log p(\mathbf{y}|\boldsymbol{\omega}, \mathbf{x}, \mathcal{S}) - \log p(\boldsymbol{\omega}|\mathbf{x}, \mathcal{S}) \right]\\
&= \log p(\mathbf{y}|\mathbf{x}, \mathcal{S}) - \mathbb{E}_{q_\phi(\boldsymbol{\omega}|\mathcal{S})} \left[\log p(\mathbf{y}|\boldsymbol{\omega}, \mathbf{x}, \mathcal{S}) \right] + D_{\mathrm{KL}}[q_\phi(\boldsymbol{\omega}|\mathcal{S})||p(\boldsymbol{\omega}|\mathbf{x}, \mathcal{S})] \geq 0.
\end{aligned} \tag{5}$$

Therefore, the lower bound of the $\log p(\mathbf{y}|\mathbf{x}, \mathcal{S})$ is

$$\log p(\mathbf{y}|\mathbf{x}, \mathcal{S}) \geq \mathbb{E}_{q_\phi(\boldsymbol{\omega}|\mathcal{S})} \log \left[p(\mathbf{y}|\mathbf{x}, \mathcal{S}, \boldsymbol{\omega}) \right] - D_{\mathrm{KL}}[q_\phi(\boldsymbol{\omega}|\mathcal{S})||p(\boldsymbol{\omega}|\mathbf{x}, \mathcal{S})], \tag{6}$$

which is consistent with (4).

2 Cross attention in the prior network

In $p(\boldsymbol{\omega}|\mathbf{x}, \mathcal{S})$, both $\mathbf{x}$ and $\mathcal{S}$ are inputs of the prior network. In order to effectively integrate the two conditions, we adopt the cross attention [2] between $\mathbf{x}$ and each element in $\mathcal{S}$. In our case, we have the key-value matrices $K = V \in \mathbb{R}^{C \times d}$, where d is the dimension of the feature representation, and C is the number of categories in the support set. We adopt the instance pooling by taking the average of samples in each category when the shot number $k > 1$.

For the query $Q_i = \mathbf{x} \in \mathbb{R}^d$, the Laplace kernel returns attentive representation for $\mathbf{x}$:

$$\textbf{Laplace}(Q_i, K, V) := W_i V \in \mathbb{R}^d, \quad W_i := \mathrm{softmax}(-\left\| Q_i - K_{j\cdot} \right\|_1)_{j=1}^C \tag{7}$$

The prior network takes the attentive representation as the input.

- *Y. Du, X. Zhen and C. Snoek are with the University of Amsterdam, Amsterdam, the Netherlands.*
- *J. Xu is with Nankai University, Tianjin, China.*
- *H. Sun, and Y. Yin are with the School of Software, Shandong University, China.*
- *X. Zhen and L. Shao are with the Inception Institute of Artificial Intelligence, UAE.*

- *A preliminary version of this work appeared in ICML 2020 [1].*

Manuscript received , 2021; revised , 2021.

3 More experimental details

We train all models using the Adam optimizer [3] with a learning rate of 0.0001. The other training setting and network architecture for regression and classification on three datasets are different as follows.

3.1 Inference networks

The architecture of the inference network with vanilla LSTM for the regression task is in Table 1. The architecture of the inference network with bidirectional LSTM for the regression task is in Table 2. For few-shot classification tasks, all models share the same architecture with vanilla LSTM, as in Table 3, For few-shot classification tasks, all models share the same architecture with bidirectional LSTM, as in Table 4.

3.2 Prior networks

The architecture of the prior network for the regression task is in Table 5. For few-shot classification tasks, all models share the same architecture, as in Table 6.

3.3 Feature embedding networks

Regression. The fully connected architecture for regression tasks is shown in Table 7. We train all three models (3-shot, 5-shot, 10-shot) over a total of $20,000$ iterations, with 6 episodes per iteration.

Classification. The CNN architectures for Omniglot, CIFAR-FS, and *mini*ImageNet are shown in Table 8, 9, and 10. The difference of feature embedding architectures for different datasets is due the different image sizes.

TABLE 1
The inference network $\phi(\cdot)$ based on the vanilla LSTM used for regression.

Output size	Layers
40	Input samples feature
40	fully connected, ELU
40	fully connected, ELU
40	LSTM cell, Tanh to μ_w, $\log \sigma_w^2$

TABLE 2
The inference network $\phi(\cdot)$ based on the bidirectional LSTM for regression.

Output size	Layers
80	Input samples feature
40	fully connected, ELU
40	fully connected, ELU
40	LSTM cell, Tanh to μ_w, $\log \sigma_w^2$

TABLE 3
The inference network $\phi(\cdot)$ based on the vanilla LSTM for Omniglot, *mini*ImageNet, CIFAR-FS.

Output size	Layers
$k \times 256$	Input feature
256	instance pooling
256	fully connected, ELU
256	fully connected, ELU
256	fully connected, ELU
256	LSTM cell, tanh to μ_w, $\log \sigma_w^2$

3.4 Other settings

The settings including the iteration numbers and the batch sizes are different on different datasets. The detailed information is given in Table 11.

TABLE 4
The inference network $\phi(\cdot)$ based on the bidirectional LSTM for Omniglot, *mini*ImageNet, CIFAR-FS.

Output size	**Layers**
$k \times 512$	Input feature
256	instance pooling
256	fully connected, ELU
256	fully connected, ELU
256	fully connected, ELU
256	LSTM cell, tanh to μ_w, $\log \sigma_w^2$

TABLE 5
The prior network for regression.

Output size	**Layers**
40	fully connected, ELU
40	fully connected, ELU
40	fully connected to μ_w, $\log \sigma_w^2$

TABLE 6
The prior network for Omniglot, *mini*ImageNet, CIFAR-FS

Output size	**Layers**
256	Input query feature
256	fully connected, ELU
256	fully connected, ELU
256	fully connected to μ_w, $\log \sigma_w^2$

TABLE 7
The fully connected network $\psi(\cdot)$ used for regression.

Output size	**Layers**
1	Input training samples
40	fully connected, RELU
40	fully connected, RELU

TABLE 8
The CNN architecture $\psi(\cdot)$ for Omniglot.

Output size	Layers
28×28×1	Input images
14×14×64	*conv2d* (3×3, stride=1, SAME, RELU), dropout 0.9, *pool* (2×2, stride=2, SAME)
7×7×64	*conv2d* (3×3, stride=1, SAME, RELU), dropout 0.9, *pool* (2×2, stride=2, SAME)
4×4×64	*conv2d* (3×3, stride=1, SAME, RELU), dropout 0.9, *pool* (2×2, stride=2, SAME)
2×2×64	*conv2d* (3×3, stride=1, SAME, RELU), dropout 0.9, *pool* (2×2, stride=2, SAME)
256	flatten

TABLE 9
The CNN architecture $\psi(\cdot)$ for CIFAR-FS

Output size	Layers
32×32×3	Input images
16×16×64	*conv2d* (3×3, stride=1, SAME, RELU), dropout 0.5, *pool* (2×2, stride=2, SAME)
8×8×64	*conv2d* (3×3, stride=1, SAME, RELU), dropout 0.5, *pool* (2×2, stride=2, SAME)
4×4×64	*conv2d* (3×3, stride=1, SAME, RELU), dropout 0.5, *pool* (2×2, stride=2, SAME)
2×2×64	*conv2d* (3×3, stride=1, SAME, RELU), dropout 0.5, *pool* (2×2, stride=2, SAME)
256	flatten

TABLE 10
The CNN architecture $\psi(\cdot)$ for *mini*ImageNet

Output size	Layers
$84\times84\times3$	Input images
$42\times42\times64$	*conv2d* (3×3, stride=1, SAME, RELU), dropout 0.5, *pool* (2×2, stride=2, SAME)
$21\times21\times64$	*conv2d* (3×3, stride=1, SAME, RELU), dropout 0.5, *pool* (2×2, stride=2, SAME)
$10\times10\times64$	*conv2d* (3×3, stride=1, SAME, RELU), dropout 0.5, *pool* (2×2, stride=2, SAME)
$5\times5\times64$	*conv2d* (3×3, stride=1, SAME, RELU), dropout 0.5, *pool* (2×2, stride=2, SAME)
$2\times2\times64$	*conv2d* (3×3, stride=1, SAME, RELU), dropout 0.5, *pool* (2×2, stride=2, SAME)
256	flatten

TABLE 11
The iteration numbers and batch sizes on different datasets.

Dataset	**Iteration**	**Batch size**
Regression	$20,000$	25
Omniglot	$100,000$	6
CIFAR-FS	$200,000$	8
*mini*ImageNet	$150,000$	8

REFERENCES

[1] X. Zhen, H. Sun, Y. Du, J. Xu, Y. Yin, L. Shao, and C. G. M. Snoek, "Learning to learn kernels with variational random features," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 409–11 419.
[2] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh, "Attentive neural processes," in *International Conference on Learning Representations*, 2019.
[3] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.