Supplementary Materials for
"Learning to Learn Kernels with Variational Random Features"

Xiantong Zhen, Haoliang Sun, Yingjun Du, Jun Xu, Yilong Yin, Ling Shao, Cees Snoek

A. Derivations of the ELBO

For a single task, we begin with maximizing log-likelihood of the conditional distribution \( p(y|x, S) \) to derive the ELBO of MetaVRF. By leveraging Jensen’s inequality, we have the following steps as

\[
\log p(y|x, S) = \log \int p(y|x, S, \omega)p(\omega|x, S)d\omega
\]

\[
= \log \int p(y|x, S, \omega)p(\omega|x, S)\frac{q_\phi(\omega|S)}{q_\phi(\omega|S)}d\omega
\]

\[
\geq \int \log \left[ \frac{p(y|x, S, \omega)p(\omega|x, S)}{q_\phi(\omega|S)} \right] q_\phi(\omega|S)d\omega
\]

\[
= \mathbb{E}_{q_\phi(\omega|S)} \log [p(y|x, S, \omega)] - D_{KL}[q_\phi(\omega|S)||p(\omega|x, S)].
\]

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

\[
D_{KL}[q_\phi(\omega|S)||p(\omega|y, x, S)] = \mathbb{E}_{q_\phi(\omega|S)} [\log q_\phi(\omega|S) - \log p(\omega|y, x, S)]
\]

\[
= \mathbb{E}_{q_\phi(\omega|S)} \left[ \log q_\phi(\omega|S) - \log \frac{p(y|\omega, x, S)p(\omega|x, S)}{p(y|x, S)} \right]
\]

\[
= \log p(y|x, S) + \mathbb{E}_{q_\phi(\omega|S)} [\log q_\phi(\omega|S) - \log p(y|\omega, x, S) - \log p(\omega|x, S)]
\]

\[
= \log p(y|x, S) - \mathbb{E}_{q_\phi(\omega|S)} [\log p(y|\omega, x, S)] + D_{KL}[q_\phi(\omega|S)||p(\omega|x, S)] \geq 0.
\]

Therefore, the lower bound of the log likelihood \( \log p(y|x, S) \) is

\[
\log p(y|x, S) \geq \mathbb{E}_{q_\phi(\omega|S)} \log [p(y|x, S, \omega)] - D_{KL}[q_\phi(\omega|S)||p(\omega|x, S)],
\]

which is consistent with (4).

B. Cross attention in the prior network

In \( p(\omega|x, S) \), both \( x \) and \( S \) are inputs of the prior network. In order to effectively integrate the two conditions, we adopt the cross attention (Kim et al., 2019) between \( x \) and each element in \( S \). 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 \( Q_i = x \in \mathbb{R}^d \), the Laplace kernel returns attentive representation for \( x \):

\[
\text{Laplace}(Q_i, K, V) := W_i V \in \mathbb{R}^d, \quad W_i := \text{softmax}(-\|Q_i - K_j\|_1)_j=1^C
\]

The prior network takes the attentive representation as the input.
C. More experimental details

We train all models using the Adam optimizer (Kingma & Ba, 2014) 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.

C.1. Inference networks

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

C.2. Prior networks

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

C.3. Feature embedding networks

Regression. The fully connected architecture for regression tasks is shown in Table D.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 miniImageNet are shown in Table D.8, D.9, and D.10. The difference of feature embedding architectures for different datasets is due the different image sizes.

| Table C.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 $\mu_w, \log \sigma^2_w$ |

| Table C.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 $\mu_w, \log \sigma^2_w$ |

| Table C.3. The inference network $\phi(\cdot)$ based on the vanilla LSTM for Omniglot, miniImageNet, CIFAR-fs. |
|---|---|
| Output size | Layers |
| $k \times 256$ | Input feature |
| 256 | instance pooling |
| 256 | fully connected, ELU |
| 256 | fully connected, ELU |
| 256 | LSTM cell, tanh to $\mu_w, \log \sigma^2_w$ |

D. Few-shot classification datasets

Omniglot (Lake et al., 2015) is a benchmark of few-shot learning that contain 1623 handwritten characters (each with 20 examples). All characters are grouped in 50 alphabets. For fair comparison against the state of the arts, we follow the same
Table C.4. The inference network $\phi(\cdot)$ based on the bidirectional LSTM for Omniglot, miniImageNet, CIFAR-FS.

<table>
<thead>
<tr>
<th>Output size</th>
<th>Layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k \times 512$</td>
<td>Input feature</td>
</tr>
<tr>
<td>256</td>
<td>instance pooling</td>
</tr>
<tr>
<td>256</td>
<td>fully connected, ELU</td>
</tr>
<tr>
<td>256</td>
<td>fully connected, ELU</td>
</tr>
<tr>
<td>256</td>
<td>fully connected, ELU</td>
</tr>
<tr>
<td>256</td>
<td>LSTM cell, tanh to $\mu_w$, $\log \sigma^2_w$</td>
</tr>
</tbody>
</table>

Table C.5. The prior network for regression.

<table>
<thead>
<tr>
<th>Output size</th>
<th>Layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>fully connected, ELU</td>
</tr>
<tr>
<td>40</td>
<td>fully connected, ELU</td>
</tr>
<tr>
<td>40</td>
<td>fully connected to $\mu_w$, $\log \sigma^2_w$</td>
</tr>
</tbody>
</table>

data split and pre-processing used in Vinyals et al. (2016). The training, validation, and testing are composed of a random split of $[1100, 200, 423]$. The dataset is augmented with rotations of 90 degrees, which results in 4000 classes for training, 400 for validation, and 1292 for testing. The number of examples is fixed as 20. All images are resized to $28 \times 28$. For a $C$-way, $k$-shot task at training time, we randomly sample $C$ classes from the 4000 classes. Once we have $C$ classes, $(k + 15)$ examples of each are sampled. Thus, there are $C \times k$ examples in the support set and $C \times 15$ examples in the query set. The same sampling strategy is also used in validation and testing.

miniImageNet (Vinyals et al., 2016) is a challenging dataset constructed from ImageNet (Russakovsky et al., 2015), which comprises a total of 100 different classes (each with 600 instances). All these images have been downsampled to $84 \times 84$. We use the same splits of Ravi & Larochelle (2017), where there are $[64, 16, 20]$ classes for training, validation and testing. We use the same episodic manner as Omniglot for sampling.

CIFAR-FS (CIFAR100 few-shots) (Bertinetto et al., 2019) is adapted from the CIFAR-100 dataset (Krizhevsky et al., 2009) for few-shot learning. Recall that in the image classification benchmark CIFAR-100, there are 100 classes grouped into 20 superclasses (each with 600 instances). CIFAR-FS use the same split criteria $(64, 16, 20)$ with which miniImageNet has been generated. The resolution of all images is $32 \times 32$.

D.1. Other settings

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

E. More results on few-shot regression

We provide more experimental results for the tasks of few-shot regression in Figure E.1. The proposed MetaVRF again performs much better than regular random Fourier features (RFFs) and the MAML method.
Table C.6. The prior network for Omniglot, miniImageNet, CIFAR-fs

<table>
<thead>
<tr>
<th>Output size</th>
<th>Layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>Input query feature</td>
</tr>
<tr>
<td>256</td>
<td>fully connected, ELU</td>
</tr>
<tr>
<td>256</td>
<td>fully connected, ELU</td>
</tr>
<tr>
<td>256</td>
<td>fully connected to $\mu_w, \log \sigma_w^2$</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Output size</th>
<th>Layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Input training samples</td>
</tr>
<tr>
<td>40</td>
<td>fully connected, RELU</td>
</tr>
<tr>
<td>40</td>
<td>fully connected, RELU</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Output size</th>
<th>Layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>28x28x1</td>
<td>Input images</td>
</tr>
<tr>
<td>14x14x64</td>
<td>conv2d (3x3, stride=1, SAME, RELU), dropout 0.9, pool (2x2, stride=2, SAME)</td>
</tr>
<tr>
<td>7x7x64</td>
<td>conv2d (3x3, stride=1, SAME, RELU), dropout 0.9, pool (2x2, stride=2, SAME)</td>
</tr>
<tr>
<td>4x4x64</td>
<td>conv2d (3x3, stride=1, SAME, RELU), dropout 0.9, pool (2x2, stride=2, SAME)</td>
</tr>
<tr>
<td>2x2x64</td>
<td>conv2d (3x3, stride=1, SAME, RELU), dropout 0.9, pool (2x2, stride=2, SAME)</td>
</tr>
<tr>
<td>256</td>
<td>flatten</td>
</tr>
</tbody>
</table>

Table D.9. The CNN architecture $\psi(\cdot)$ for CIFAR-fs

<table>
<thead>
<tr>
<th>Output size</th>
<th>Layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>32x32x3</td>
<td>Input images</td>
</tr>
<tr>
<td>16x16x64</td>
<td>conv2d (3x3, stride=1, SAME, RELU), dropout 0.5, pool (2x2, stride=2, SAME)</td>
</tr>
<tr>
<td>8x8x64</td>
<td>conv2d (3x3, stride=1, SAME, RELU), dropout 0.5, pool (2x2, stride=2, SAME)</td>
</tr>
<tr>
<td>4x4x64</td>
<td>conv2d (3x3, stride=1, SAME, RELU), dropout 0.5, pool (2x2, stride=2, SAME)</td>
</tr>
<tr>
<td>2x2x64</td>
<td>conv2d (3x3, stride=1, SAME, RELU), dropout 0.5, pool (2x2, stride=2, SAME)</td>
</tr>
<tr>
<td>256</td>
<td>flatten</td>
</tr>
</tbody>
</table>

Table D.10. The CNN architecture $\psi(\cdot)$ for miniImageNet

<table>
<thead>
<tr>
<th>Output size</th>
<th>Layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>84x84x3</td>
<td>Input images</td>
</tr>
<tr>
<td>42x42x64</td>
<td>conv2d (3x3, stride=1, SAME, RELU), dropout 0.5, pool (2x2, stride=2, SAME)</td>
</tr>
<tr>
<td>21x21x64</td>
<td>conv2d (3x3, stride=1, SAME, RELU), dropout 0.5, pool (2x2, stride=2, SAME)</td>
</tr>
<tr>
<td>10x10x64</td>
<td>conv2d (3x3, stride=1, SAME, RELU), dropout 0.5, pool (2x2, stride=2, SAME)</td>
</tr>
<tr>
<td>5x5x64</td>
<td>conv2d (3x3, stride=1, SAME, RELU), dropout 0.5, pool (2x2, stride=2, SAME)</td>
</tr>
<tr>
<td>2x2x64</td>
<td>conv2d (3x3, stride=1, SAME, RELU), dropout 0.5, pool (2x2, stride=2, SAME)</td>
</tr>
<tr>
<td>256</td>
<td>flatten</td>
</tr>
</tbody>
</table>

Table D.11. The iteration numbers and batch sizes on different datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Iteration</th>
<th>Batch size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>20,000</td>
<td>25</td>
</tr>
<tr>
<td>Omniglot</td>
<td>100,000</td>
<td>6</td>
</tr>
<tr>
<td>CIFAR-fs</td>
<td>200,000</td>
<td>8</td>
</tr>
<tr>
<td>miniImageNet</td>
<td>150,000</td>
<td>8</td>
</tr>
</tbody>
</table>
Figure E.1. More results of few-shot regression. (--- MetaVRF with bi-LSTM; - MetaVRF with LSTM; -- MetaVRF w/o LSTM; —- MAML; —- Ground Truth; ▲ Support Samples.)
References


