Learning to Learn with Variational Information Bottleneck for Domain Generalization

Yingjun Du[1][0000−0001−7537−6457], Jun Xu[2], Huan Xiong[4], Qiang Qiu[5], Xiantong Zhen[1,3], Cees G. M. Snoek[1], and Ling Shao[3,4]

[1] AIM Lab, University of Amsterdam, The Netherlands
[2] College of Computer Science, Nankai University, China
[3] Inception Institute of Artificial Intelligence, Abu Dhabi, UAE
[4] Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE
[5] Electrical and Computer Engineering, Duke University, USA
{y.du, x.zhen, cgmsnoek}@uva.nl, nankaimathxujun@gmail.com, huan.xiong@mbzuai.ac.ae, qiang.qiu@duke.edu, ling.shao@ieee.org

1 Algorithms of MetaVIB for Training

We describe the detailed algorithm for training MetaVIB as following Algorithm 1:

2 Learning Architecture

To better clearly understand our proposed MetaVIB, we draw a concise architecture diagram in Fig. 1.

3 Training Details

During the training, we use the Adam [1] optimizer, and set the learning rate as 10^{-4}. In each training batch, we randomly select three domains including two meta-train domains and one meta-test domain. In each domain, we choose 256 samples, and the batch size is 256×3. The iteration number is set as $25,000$. The model with the highest validation accuracy is employed to evaluate the test set from the meta-test domain.

4 Influence of information bottleneck size β

We report Influence of information bottleneck size β on the VLCS and Rotated MNIST in Tables 1 and 2. For the VLCS, MetaVIB obtains best results for $\beta = 0.01$, while for the Rotated MNIST, MetaVIB gets best results for $\beta = 0.001$.

Algorithm 1 Learning to Learn with Variational Information Bottleneck for Domain Generalization

Input: Training data $\mathcal{S}$ of K source domains; learning rate λ; the number of iteration N_{iter}.
Initialize the parameters $\boldsymbol{\Theta} = \{\theta, \phi_1, \phi_2\}$ of the model including the feature extraction network $h_\theta(\cdot)$ and the inference networks $g_{\phi_1}(\cdot)$ and $g_{\phi_2}(\cdot)$.
for *iter* in N_{iter} **do**
  $D^t \leftarrow \text{RANDOMSAMPLE}(\{1, \cdots, K\}, t)$;
  $D^s \leftarrow \{1, \cdots, K\} \setminus D^t$;
  Sample $\{(\mathbf{x}_m^s, \mathbf{y}_m^s)\}_{m=1}^M \sim D^s$; $\{(\mathbf{x}_n^t, \mathbf{y}_n^t)\}_{n=1}^N \sim \mathcal{D}^t$;
  for c in $1:C$ **do**
    $\overline{\mathbf{h}}_c^s = \frac{1}{M_c} \sum_{i=1}^{M_c} h_\theta(\mathbf{x}_{i,c}^s)$; $\boldsymbol{\mu}_{\boldsymbol{c}}^{\psi}, \boldsymbol{\sigma}_{\boldsymbol{c}}^{\psi} = g_{\phi_1}(\overline{\mathbf{h}}_c^s)$;
    $\psi_c \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{c}}^{\psi}, \text{diag}((\boldsymbol{\sigma}_{\boldsymbol{c}}^{\psi})^2)$;
  end for
  $\psi = [\psi_1, \cdots, \psi_c, \cdots, \psi_C]$;
  for c in $1:C$ **do**
    $\overline{\mathbf{h}}_c^s = \frac{1}{M_c} \sum_{i=1}^{M_c} h_\theta(\mathbf{x}_{i,c}^s)$; $\boldsymbol{\mu}_c^s, \boldsymbol{\sigma}_c^s = g_{\phi_2}(\overline{\mathbf{h}}_c^s)$;
    $\mathbf{z}_c \sim \mathcal{N}(\boldsymbol{\mu}_c^s, \text{diag}((\boldsymbol{\sigma}_c^s)^2))$;
    $\boldsymbol{\mu}_{j,c}^t, \boldsymbol{\sigma}_{j,c}^t = g_{\phi_2}(h_\theta(\mathbf{x}_{j,c}^t))$;
    $\mathbf{z}_{j,c} \sim \mathcal{N}(\boldsymbol{\mu}_{j,c}^t, \text{diag}((\boldsymbol{\sigma}_{j,c}^t)^2))$;
    $\mathcal{L}_c = \sum_{(\mathbf{x}_{j,c}^t, \mathbf{y}_c^t)} \left[-\psi_y \cdot \mathbf{z}_{j,c} + \log(\sum_{c=1}^{C} e^{\psi_c \cdot \mathbf{z}_{j,c}}) \right] + \beta D_{\text{KL}}(q(\mathbf{z}_c|\overline{\mathbf{h}}_c)||p(\mathbf{z}_{j,c}|h_\theta(\mathbf{x}_{j,c}^t)))$;
  end for
  Update parameters: $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \lambda \sum_{c=1}^{C} \nabla_\Theta \mathcal{L}_c$.
end for

5 Influence of the number of Monte Carlo Influence of the number of Monte Carlo samples

We use Monte Carlo sampling to draw samples from$p(\mathbf{z}|\mathbf{x})$ for $\mathbf{z}$. We report varying sample number L_z on PACS in the Table 3. Our method achieves inferior results with $L_z = 1$; performs consistently better with $L_z = 5, 10$, converges at $L_z = 10$ and becomes worse when $L_z = 50, 100$. So in our experiments, we set $L_z = 10$ and we averaged over 20 runs on the test domain. The variance reflects the error caused by Monte Carlo sampling in each test experiment.

6 Network Architectures

6.1 Feature Embedding Network

The feature extraction network for **PACS**, **VLCS** is shown in Table 4, the feature extraction network for **Rotated MNIST** is shown in Table 5.

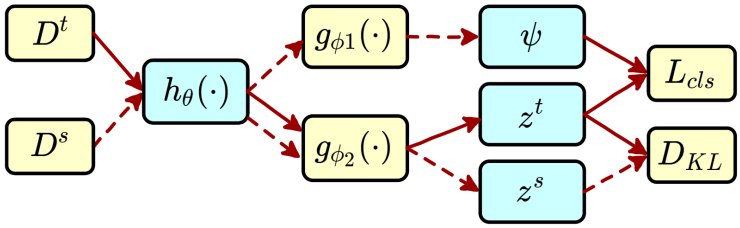

Fig. 1. Architecture diagram. $h_\theta(\cdot)$ is the feature extraction network; $g_{\phi_1}(\cdot)$ is the inference network to generate the distribution of classifier parameters ψ; $g_{\phi_2}(\cdot)$ is the inference network to generate the latent distribution of z; L_{cls} is the cross-entropy loss. Solid (Dashed) line represents the direction of data flow in the meta-test domain D^t (meta-train domain D^s).

Table 1. Influence of information bottleneck size β on domain generalization for VLCS.

	VOC2007	**LabelMe**	**Caltech-101**	**SUN09**	**Mean**
$\beta = 1$	67.15±0.31	60.32±0.37	94.83±0.25	65.02±0.23	71.83
$\beta = 0.1$	68.93±0.24	61.31±0.18	95.98±0.21	67.05±0.21	73.32
$\beta = 0.01$	70.28±0.34	62.66±0.24	97.37±0.33	67.85±0.27	74.54
$\beta = 0.001$	68.47±0.35	61.17±0.27	95.35±0.23	66.90±0.25	72.97

6.2 Inference Network

The architecture of the inference network $g_{\phi_1}(\cdot)$ for **PACS**, **VLCS** is in Table 6, the architecture of the inference network $g_{\phi_1}(\cdot)$ for **Rotated MNIST** is in Table 7.

The architecture of the inference network $g_{\phi_2}(\cdot)$ for **PACS**, **VLCS** is in Table 8, the architecture of the inference network $g_{\phi_2}(\cdot)$ for **Rotated MNIST** is in Table 9.

7 Prediction Uncertainty Analysis

Since the data follows distinct distribution between seen and unseen domains, uncertainty is inevitable during the prediction stage on the unseen domains, to which no data is accessible in the learning stage. To deal with the prediction uncertainty, we model parameters of classifiers shared across domains as

Table 2. Influence of information bottleneck size β on domain generalization for Rotated MNIST.

	$\mathbf{M_{0^\circ}}$	$\mathbf{M_{15^\circ}}$	$\mathbf{M_{30^\circ}}$	$\mathbf{M_{45^\circ}}$	$\mathbf{M_{60^\circ}}$	$\mathbf{M_{75^\circ}}$	**Mean**
$\beta = 1$	89.13±0.24	98.01±0.21	97.38±0.18	97.32±0.20	98.13±0.28	88.72±0.13	94.78
$\beta = 0.1$	90.35±0.31	98.17±0.21	98.82±0.34	98.18±0.31	98.73±0.29	89.94±0.17	95.69
$\beta = 0.01$	91.05±0.19	99.35±0.03	99.10±0.31	99.38±0.18	99.27±0.18	91.94±0.47	96.68
$\beta = 0.001$	91.28±0.21	99.90±0.02	99.29±0.11	99.78±0.10	99.57±0.13	92.75±0.31	97.08

Table 3. Influence of the number of Monte Carlo samples L_z on domain generalization for PACS. MetaVIB obtains best results for $L_z = 10$.

	Photo	**Art painting**	**Cartoon**	**Sketch**	**Mean**
$L_z = 1$	89.32±0.41	69.17±0.37	70.37±0.27	62.84±0.45	72.93
$L_z = 5$	90.11±0.17	70.26±0.38	71.93±0.21	63.45±0.46	73.94
$L_z = 10$	91.93±0.23	71.94±0.34	73.17±0.21	65.94±0.24	75.74
$L_z = 50$	91.82±0.25	71.74±0.32	73.37±0.17	66.01±0.38	75.73
$L_z = 100$	91.71±0.35	71.87±0.37	73.09±0.27	65.81±0.48	75.62

probabilistic distributions that we infer from the data of the seen domains. The probabilistic modeling enables us to better handle the prediction uncertainty on previously unseen domains.

In order to demonstrate that the proposed probabilistic modeling can handle prediction uncertainty, we conduct an extra set of experiments as follows:

We shown more success and failure cases in Fig. 2 and show the corresponding prediction probabilities of using different sampled classifiers ψ for each category of the image in Fig. 3-10. ψ_μ indicates the mean value of the classifier. From Fig. 3-10, we can see that different ψ can produce different prediction probabilities to each category. Specially, for the fourth image of success cases, the final result of the classification is *giraffe*. However, the classifiers ψ_1 and ψ_2, our model predicts a higher prediction probability of *horse* than *giraffe* as shown in Fig. 6. For the fourth image of failure cases, the image is classified as *dog*, but that the prediction probability of *elephant* is higher than that of *dog* by using classifiers ψ_4 as shown in Fig. 10. Although the final prediction result of our model is incorrect, some of sampled classifiers can still make correct predictions.

(a) Success cases (b) Failure cases

Fig. 2. Success and failure cases of MetaVIB. The numbers associated with each image are the top two prediction probabilities of MetaVIB, with ground truth labels in red.

Table 4. The feature extraction network $h_\theta(\cdot)$ for **PACS**, **VLCS**

Feature Extraction Network : $h_\theta(\cdot)$

Output size	**Layers**
$227 \times 227 \times 3$	Input image
$27 \times 27 \times 96$	conv2d (11×11, stride 4, SAME, RELU), pool (3×3, stride 2, VALID)
$13 \times 13 \times 256$	conv2d (5×5, stride 1, SAME, RELU), pool (3×3, stride 2, VALID)
$13 \times 13 \times 384$	conv2d (3×3, stride 1, SAME, RELU)
$13 \times 13 \times 384$	conv2d (3×3, stride 1, SAME, RELU)
$6 \times 6 \times 256$	conv2d (3×3, stride 1, SAME, RELU) , pool (3×3, stride 2, VALID)
4096	fully connected, RELU, dropout
4096	fully connected, RELU

Table 5. The feature extraction network $h_\theta(\cdot)$ for **Rotated MNIST**

Feature Extraction Network : $h_\theta(\cdot)$

Output size	**Layers**
$28 \times 28 \times 1$	Input image
$14 \times 14 \times 32$	conv2d (3×3, stride 1, SAME, RELU), pool (3×3, stride 2, VALID)
$7 \times 7 \times 32$	conv2d (3×3, stride 1, SAME, RELU), pool (3×3, stride 2, VALID)
256	fully connected, RELU

References

1. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

Table 6. Inference network $g_{\phi_1}(\cdot)$ used for **PACS**, **VLCS**.

Inference Network: $g_{\phi_1}(\cdot)$	
Output size	**Layers**
$k \times 4096$	Input feature
4096	instance pooling
1024	fully connected, ELU
1024	fully connected, ELU
1024	fully connected to μ_c^{ψ}, $\log(\sigma_c^{\psi})^2$

Table 7. Inference network $g_{\phi_1}(\cdot)$ used for **Rotated MNIST**.

Inference Network: $g_{\phi_1}(\cdot)$	
Output size	**Layers**
$k \times 256$	Input feature
256	instance pooling
256	fully connected, ELU
256	fully connected, ELU
256	fully connected to μ_c^{ψ}, $\log(\sigma_c^{\psi})^2$

Table 8. Inference network $g_{\phi_2}(\cdot)$ used for **PACS**, **VLCS**.

Inference Network: $g_{\phi_2}(\cdot)$	
Output size	**Layers**
$k \times 4096$	Input feature
4096	instance pooling
1024	fully connected, ELU
1024	fully connected, ELU
1024	fully connected to μ_c, $\log(\sigma_c)^2$

Table 9. Inference network $g_{\phi_2}(\cdot)$ used for **Rotated MNIST**.

Inference Network: $g_{\phi_2}(\cdot)$	
Output size	**Layers**
$k \times 256$	Input feature
256	instance pooling
256	fully connected, ELU
256	fully connected, ELU
256	fully connected to μ_c, $\log(\sigma_c)^2$

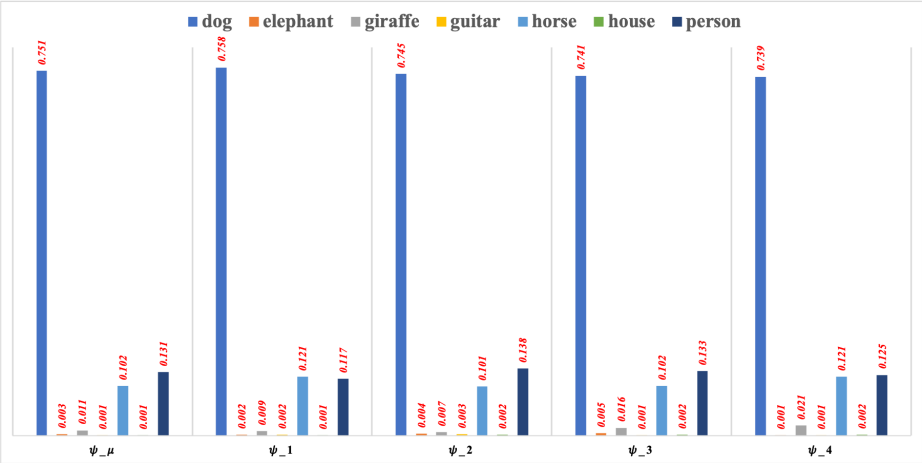

Fig. 3. The prediction probability of the different sampled classifier ψ for each category (the first image of the success cases in Fig. 2).

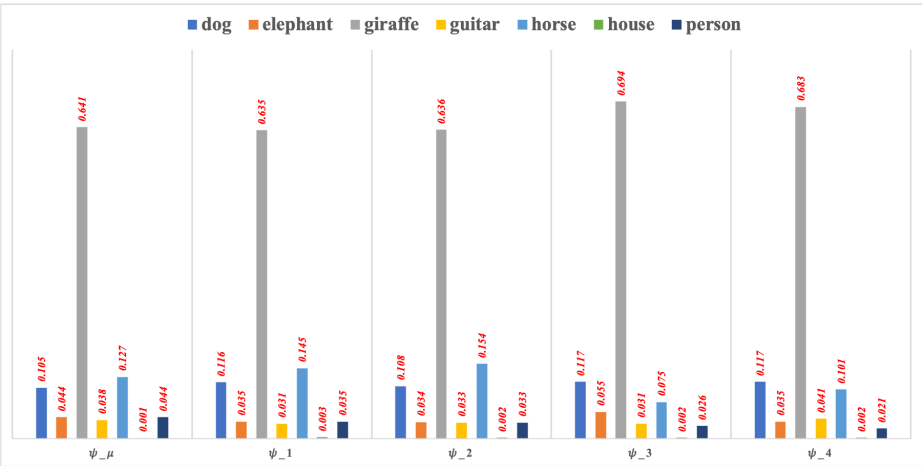

Fig. 4. The prediction probability of the different sampled classifier ψ for each category (the Second image of the success cases in Fig. 2).

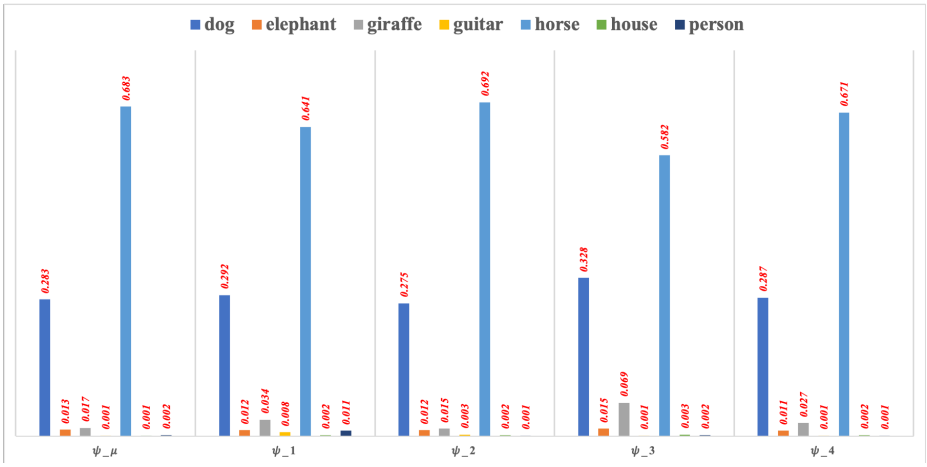

Fig. 5. The prediction probability of the different sampled classifier ψ for each category (the third image of the success cases in Fig. 2).

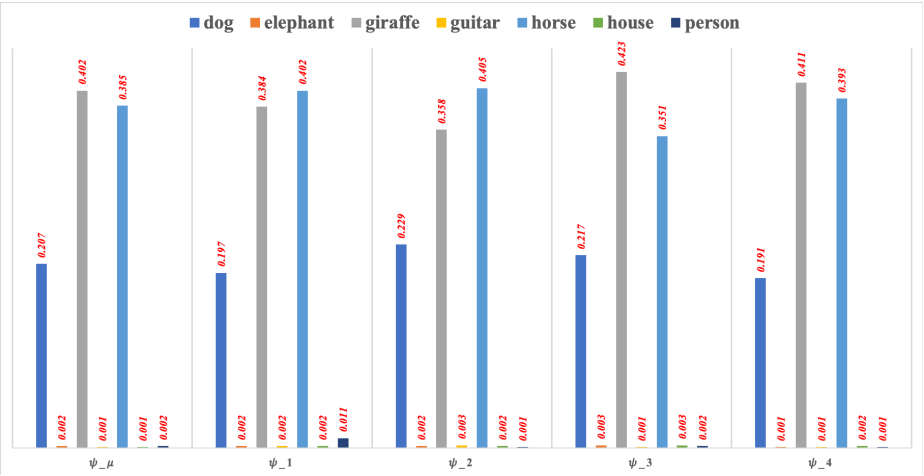

Fig. 6. The prediction probability of the different sampled classifier ψ for each category (the fourth image of the success cases in Fig. 2).

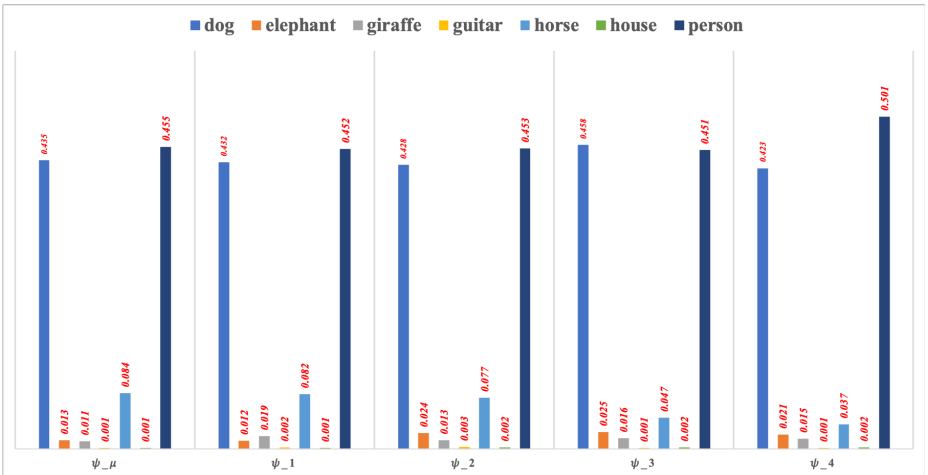

Fig. 7. The prediction probability of the different sampled classifier ψ for each category (the first image of the failure cases in Fig. 2).

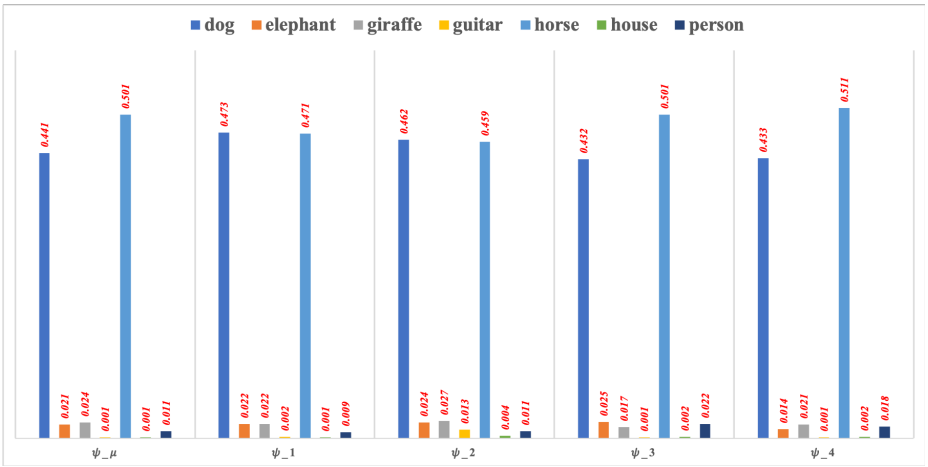

Fig. 8. The prediction probability of the different sampled classifier ψ for each category(the second image of the failure cases in Fig. 2).

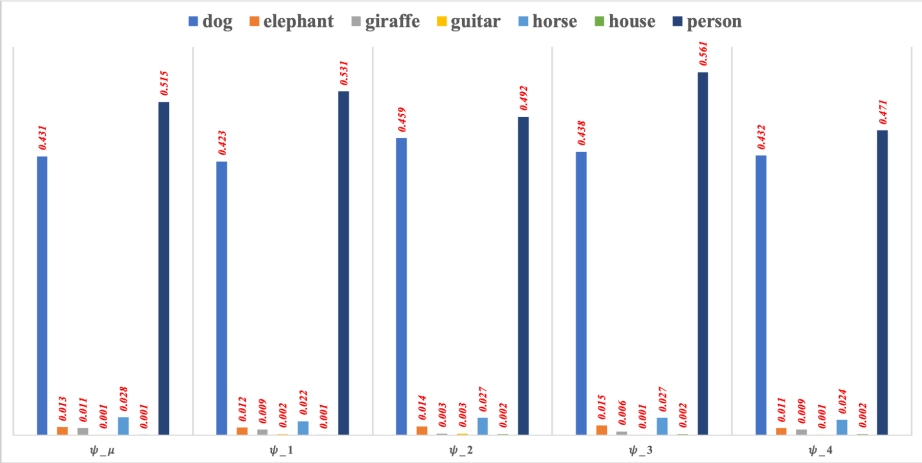

Fig. 9. The prediction probability of the different sampled classifier ψ for each category(the third image of the failure cases in Fig. 2).

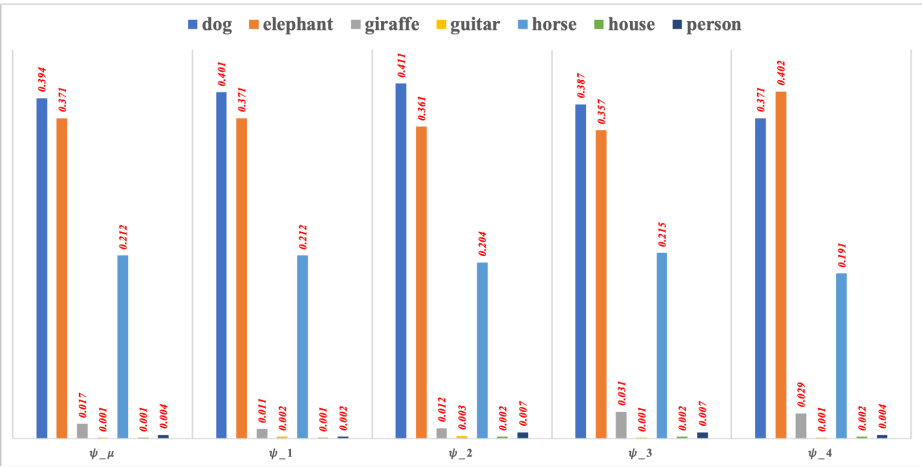

Fig. 10. The prediction probability of the different sampled classifier ψ for each category (the fourth image of the failure cases in Fig. 2).