Supplementary to "Reinforced Cross-Domain Knowledge Distillation on Time Series Data"

Anonymous Author(s)
Affiliation
Address
`email`

1 Uncertainty Estimation with Monte Carlo Dropout Method

Given an input data set $X = \{x_1, ..., x_n\}$ and the respective outputs $Y = \{y_1, ..., y_n\}$, the conventional machine learning methods intend to find an optimal model $\Phi(x; \boldsymbol{\theta})$, which is parameterized with $\boldsymbol{\theta}$, to map the input X to the Y. After training, the optimal model $\Phi(x; \boldsymbol{\theta})$ will give a single point prediction for certain test sample with static $\boldsymbol{\theta}$. On the contrary, the Bayesian methods (*e.g.*, Bayesian neural networks) can generate predictive distributions instead of a single point prediction for estimating model uncertainty. With defining $\Phi(x; \boldsymbol{\theta})$ with a prior $P(\boldsymbol{\theta})$ over parameter space $\boldsymbol{\theta}$, the training objective is then turned to find an optimal posterior distribution over $\boldsymbol{\theta}$:

$$P(\boldsymbol{\theta}|X, Y) = \frac{P(Y|X, \boldsymbol{\theta})P(\boldsymbol{\theta})}{P(Y|X)}. \tag{1}$$

The prediction value of y with input x is the weighted average of model predictions over all possible sets of parameters $\boldsymbol{\theta}$ with various posterior probabilities as Eq. (2) shows.

$$\begin{aligned} P(y|x, X, Y) &= \int P(y|x, \boldsymbol{\theta})P(\boldsymbol{\theta}|X, Y)d\boldsymbol{\theta} \\ &= \mathbb{E}_{\boldsymbol{\theta} \sim P(\boldsymbol{\theta}|X,Y)}[\Phi(x; \boldsymbol{\theta})] \end{aligned} \tag{2}$$

However, the posterior distribution $P(\boldsymbol{\theta}|X, Y)$ is intractable as shown in previous works. Alternatively, Gal and Ghahramani proved that a DNN with arbitrary non-linear depth and dropout is mathematically equivalent to a Bayesian approximation of the probabilistic deep Gaussian process. They proposed a method named Monte Carlo Dropout which utilizes a dropout distribution $\hat{P}(\boldsymbol{\theta})$ to approximate $P(\boldsymbol{\theta}|X, Y)$. To be specific, for the l-th layer ($l = 1, ..., L$) in a model with total L layers, the parameter distribution $\boldsymbol{\theta}_l$ is defined as:

$$\boldsymbol{\theta}_l = \mathbf{M}_l * diag([Z_{l,i}]_{i=1}^{D_l}), \tag{3}$$

where $\mathbf{M}_l \in \mathcal{R}^{D_l \times D_{l-1}}$ is a matrix with variational parameters and $diag(\cdot)$ is an operator to map a vector to a diagonal matrix. $Z_{l,i} \sim Bernoulli(q_i)$ is independently sampled from Bernoulli distribution, where $i = 1, ..., D_{l-1}$. q_i is the probability of dropout. Subsequently, the Eq. (2) is reformulated as:

$$\mathbb{E}_{\boldsymbol{\theta} \sim \hat{P}(\boldsymbol{\theta})}[\Phi(x; \boldsymbol{\theta})] \approx \frac{1}{N} \sum_{n=1}^{N} \Phi(x; \boldsymbol{\theta}^n). \tag{4}$$

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.
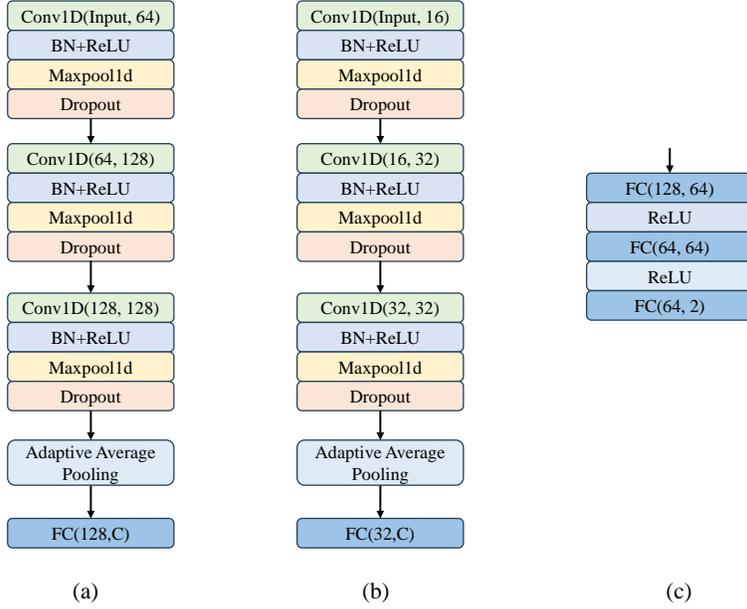
Figure 1: Network Architectures for (a) **1D-CNN** Teacher, (b) **1D-CNN** Student and (c) Domain Discriminator.

Practically, Eq. (4) means to enable the dropout of model during test phase and forward N times for each sample x_i. Furthermore, for classification tasks we employ the entropy to measure teacher's uncertainty on target data as Eq. (5) shows:

$$\mathcal{H}_i = -\sum_c (\frac{1}{N}\sum_n p(y=c|x_i,\boldsymbol{\theta}^n) log(\frac{1}{N}\sum_n p(y=c|x_i,\boldsymbol{\theta}^n))), \tag{5}$$

where $p(y=c|x_i,\boldsymbol{\theta}^n)$ represents the probability of sample belong to class c and it is the softmax outputs of teacher model $\Phi(x;\boldsymbol{\theta}^n)$ on the n-th forward pass. Intuitively, a higher value of the predictive entropy $\mathcal{H}_i$ will be obtained when all classes are predicted to have equal probabilities, which means the teacher is less confident about the specific data.

2 Model Architecture

2.1 Teacher and Student model

As illustrated in Fig. 1(a) and (b), the teacher and student model are constructed with three stacked CNN blocks as the backbone and one fully connected layer as the classifier. Each CNN block consists of a 1D convolutional layer, followed by a BatchNorm layer, an activation layer (ReLU), a 1D MaxPooling layer and a Dropout layer. Here, 'Conv1D' represents the 1D convolutional layer and the first variable in the bracket represents the number of input channels and the second one represents the number of output channels. 'BN' is a BatchNorm layer. 'FC' represents a fully connected layer. 'C' represents the number of classes.

Meanwhile, we also compared the complexity of teacher and student model as shown in Table 1. The compact student is about $15\times$ smaller in terms of number of trainable parameters and $17\times$ faster in terms of number of floating point operations (FLOPs) than the cumbersome teacher model.

2.2 Domain discriminator

Fig. 1(c) is the network architecture of domain discriminator. It consists of three linear layer followed by ReLU activation layers. The output of domain discriminator is a 2-classes probability distribution

2

Table 1: Comparison of model complexity.

	No. of trainable Parameters	No. of FLOPs
Teacher	2.01×10^5	9.17×10^5
Student	0.13×10^5	0.54×10^5
Compression Ratio	$15.46\times$	$16.98\times$

Table 2: Network Architecture for Dueling DDQN.

Layers	Dueling DDQN	
#1	FC(Input, 1024)+ReLU	
#2	NoisyFC(1024,1024)+ReLU	NoisyFC(1024,1024)+ReLU
#3	NoisyFC(1024,1) +ReLU	NoisyFC(1024, 2)+ReLU
#4	Aggregation	

43 to indicate whether the feature maps come from the teacher with source domain data as input or from
44 the student with target domain data as input.

2.3 Dueling DDQN

46 Table 2 presents the details of our dueling DDQN. The left column is the state-value estimation
47 stream and the right column is the advantage estimation column. The 'NoisyFC' represents a linear
48 layer whose weights and biases are perturbed by a parametric function of the noise. The conventional
49 linear layer can be expressed as $y = wx + b$, where $w \in \mathbb{R}^{q \times p}$, $b \in \mathbb{R}^q$ are trainable weights and
50 biases, $x \in \mathbb{R}^p$ and $y \in \mathbb{R}^q$ are the inputs and outputs, respectively. In the NoisyNets, the weights
51 and biases are reformulated as Eq. (6) and (7), respectively. Here, $\mu^w \in \mathbb{R}^{q \times p}$, $\sigma^w \in \mathbb{R}^{q \times p}$, $\mu^b \in \mathbb{R}^q$,
52 $\sigma^b \in \mathbb{R}^q$ are the trainable weights and biases. $\odot$ is the element-wise multiplication. ϵ^w and ϵ^b are the
53 factorised Gaussian noise, where each entry $\epsilon_{i,j}^w = f(\epsilon_i)f(\epsilon_j)$, $\epsilon_j^b = f(\epsilon_j)$ and $f(x) = sgn(x)\sqrt{|x|}$.
54 Adding such parametric noise to the deep reinforcement learning agent will enhance the efficiency of
55 exploration.

$$w = \mu^w + \sigma^w \odot \epsilon^w \tag{6}$$

$$b = \mu^b + \sigma^b \odot \epsilon^b \tag{7}$$

3 Additional Transfer Scenarios

57 We evaluate our proposed method on another five additional transfer scenarios on four datasets
58 as shown in Table 3 and Table 4. From above two Tables, we can sse that our proposed method
59 consistently achieves better performance than other SOTA methods, further indicating its effectiveness
60 in transferring the knowledge under the cross-domain scenarios.

Table 3: Marco F1-scores on HAR and HHAR across three independent runs.

Methods	HAR Transfer Scenarios						HHAR Transfer Scenarios					
	$18 \rightarrow 27$	$20\rightarrow5$	$24\rightarrow8$	$28\rightarrow27$	$30\rightarrow20$	Avg	$0\rightarrow2$	$5\rightarrow0$	$6\rightarrow1$	$7\rightarrow4$	$8\rightarrow3$	Avg
Teacher	98.23	90.57	97.08	100	92.21	95.62	66.56	33.25	94.47	94.99	96.68	77.19
Student-Only	98.37	48.78	77.38	61.17	76.41	72.42	61.94	27.43	69.10	77.72	80.51	63.34
KD-STDA	**100**	75.77	90.77	97.77	86.36	90.13	61.93	28.04	92.65	91.33	96.30	74.05
KA-MCD	85.22	78.03	86.14	91.19	74.28	82.97	43.90	33.35	92.32	94.27	97.02	72.17
MLD-DA	98.82	80.57	91.90	**100**	91.69	92.60	65.44	31.10	92.97	94.97	95.87	76.07
REDA	98.20	**95.05**	91.26	98.53	72.04	91.02	54.18	32.56	88.50	88.84	96.18	72.05
AAD	90.27	66.88	86.09	94.73	84.82	84.56	58.23	37.24	91.47	81.99	83.61	70.51
MobileDA	92.86	84.96	90.45	79.12	77.56	84.99	50.27	30.83	76.12	89.70	79.25	65.23
UNI-KD	**100**	94.42	**100**	92.26	87.10	94.76	62.33	**39.01**	92.89	**96.90**	96.52	77.53
Proposed	**100**	85.26	97.92	**100**	**92.21**	**95.08**	**67.27**	38.25	**94.59**	95.83	**97.40**	**78.67**

3

Table 4: Marco F1-scores on FD and SSC across three independent runs.

Methods	FD Transfer Scenarios						SSC Transfer Scenarios					
	1→0	1→3	3→0	3→1	3→2	Avg	3→19	5→15	6→2	13→17	18→12	Avg
Teacher	66.40	100	62.30	100	81.65	82.07	71.85	73.69	72.21	50.74	52.96	64.29
Student-Only	45.13	91.14	44.84	93.03	70.55	68.94	43.65	41.04	48.21	38.78	47.28	43.79
KD-STDA	47.55	93.02	51.26	99.81	76.28	73.58	61.24	66.97	67.05	43.05	49.92	57.65
KA-MCD	51.77	98.69	48.50	93.65	83.05	75.13	61.13	63.23	70.96	39.56	46.86	56.35
MLD-DA	51.98	99.67	52.14	96.01	75.62	75.08	66.23	70.30	69.33	44.22	44.13	57.65
REDA	53.65	96.21	52.48	96.60	80.85	75.96	56.09	61.96	53.59	40.50	36.26	49.68
AAD	46.42	94.65	52.09	98.65	87.11	75.78	62.75	64.81	**71.78**	44.52	49.18	58.61
MobileDA	51.71	94.92	51.17	99.86	78.51	75.23	64.16	67.67	56.74	47.50	**56.56**	58.53
UNI-KD	60.91	**99.97**	61.00	100	87.08	81.79	66.84	70.76	65.70	**50.19**	49.77	60.65
Proposed	**61.99**	99.67	**62.42**	100	**87.76**	**82.37**	**69.49**	**72.73**	67.01	49.31	52.52	**62.21**
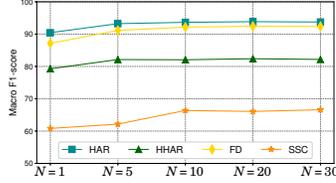
Figure 2: Sensitivity of N.
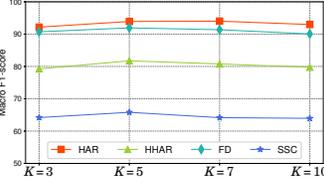
Figure 3: Sensitivity of K.
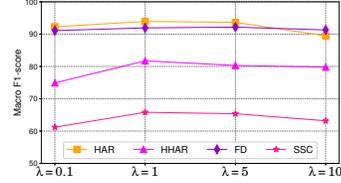
Figure 4: Sensitivity of λ.

Table 5: Sensitivity Analysis on α_1 and α_2.

Dataset	α_1=0.2 α_2=1.8	α_1=0.4 α_2=1.6	α_1=0.6 α_2=1.4	α_1=0.8 α_2=1.2	α_1=1.0 α_2=1.0	α_1=1.2 α_2=0.8	α_1=1.4 α_2=0.6	α_1=1.6 α_2=0.4	α_1=1.8 α_2=0.2
HAR	**94.68**	94.23	94.49	93.58	93.25	92.65	92.68	92.72	92.05
HHAR	**82.37**	82.35	94.49	82.10	81.59	82.11	81.04	81.25	81.34
FD	92.63	92.34	**92.87**	92.10	91.89	91.74	92.01	91.34	92.05
SSC	68.26	**68.47**	68.15	67.65	67.80	66.65	66.95	66.35	66.01

4 Sensitivity Analysis

4.1 Hyper parameter N

In our proposed method, one of the key hyper parameters is N, which is the number of teachers utilized for calculating the uncertainty on a specific sample. It relates to our reward function, thus affecting the learning process of target sample selection policy. Intuitively, the larger N will lead to more accurate estimation of teacher's uncertainty and provide more accurate reward. As illustrated in Fig. 2, student's performance is gradually increased with N but will keep at some certain level when $N \geq 10$. The larger value of N also increases the total cost in terms of training time. Empirically, $N = 5$ or $N = 10$ are appropriate, and we choose $N = 10$ in our experiments for all the datasets.

4.2 Hyper parameter K

Another hyper parameter in our proposed approach is the episodes length K for generating historical experience and we perform the analysis as illustrated in Fig. 3. From Fig. 3, we can see that our proposed method is not very sensitive to K. But a large value of K will result in longer training time. $K = 5$ is sufficient to generate informative historical experience for training the dueling DDQN.

4.3 Hyper parameter λ

Regarding hyper parameter λ which is to balance the contribution of domain confusion loss $\mathcal{L}_{DC}$ and the distillation loss $\mathcal{L}_{RKD}$, we can see from Fig. 4 that a small value of λ (*e.g.* $\lambda = 0.1$) will make the student more focus on learning domain-invariant knowledge from $\mathcal{L}_{DC}$. It will result in worse performance in datasets like **HHAR** and **SSC**. A higher value of λ will obviously enhance student's generalization capability on target domain. $\lambda \in [1, 5]$ is a suitable range based on our experiment results.

4

4.4 Hyper parameter α_1 and α_2

To limit our reward within the range of -1 to 1, α_1 and α_2 should satisfy below constrains: $\alpha_1 \in (0, 2)$, $\alpha_2 \in (0, 2)$ and $\alpha_1 + \alpha_2 = 2$. We perform grid search on four datasets as shown in Table 5. We can see that the student trained with low α_1 value and high α_2 value can achieve better performance than other configurations, indicating transferability contributes more to the final performance than uncertainty. In all of our experiments, we set $\alpha_1 = 0.2$ and $\alpha_2 = 1.8$ for all datasets for simplicity.

5