Improving the Reliability for Confidence Estimation (Supplementary)

Haoxuan Qu[1⋆], Yanchao Li[1⋆], Lin Geng Foo[1], Jason Kuen[2], Jiuxiang Gu[2], and Jun Liu[1⋆⋆]

[1] Singapore University of Technology and Design
{haoxuan_qu, lingeng_foo}@mymail.sutd.edu.sg, {yanchao_li, jun_liu}@sutd.edu.sg
[2] Adobe Research
{kuen, jigu}@adobe.com

1 Relationship between Our Framework and Train-Validation Split Scheme

In our framework, in each iteration, we first train the confidence estimator on the virtual training set (i.e. *virtual training*). After that, the trained confidence estimator is evaluated on the virtual testing set (i.e. *virtual testing*). Finally, the evaluation result (virtual testing loss $L_{v\_te}(\phi')$) computed on the virtual testing set is used as a *feedback* to the training procedure of the confidence estimator to drive it to learn more *distribution-generalizable* knowledge (i.e. via *meta-optimization*). We notice that the classical train-validation split scheme has some relationships in concept with our meta framework, as it also trains the model on the training set first, and then evaluates the model on the validation set. However, we want to highlight here that our framework with the *virtual training and testing* scheme incorporated is totally different from the classical train-validation split scheme. We explain the two reasons in more detail below.

Firstly, the classical train-validation split scheme is often used to adjust the model parameters indirectly through adjusting the hyperparameters, which lacks a mechanism to directly and automatically optimizes the model parameters, while our framework incorporates a *feedback* mechanism to directly optimize the confidence estimator parameters during training to drive the confidence estimator to learn more *distribution-generalizable* knowledge. This cannot be done by simply utilizing the train-validation split scheme.

Secondly, we *intentionally design distribution differences* between the constructed virtual training and virtual testing sets in our framework. Hence, utilizing the evaluation result computed on the virtual testing set with a different distribution from the virtual training set as the feedback, our framework further encourages the knowledge learned by the confidence estimator during training to be more *distribution-generalizable*.

⋆ Both authors contributed equally to the work.
⋆⋆ Corresponding Author

Due to the effectiveness of our framework, during experiments, we empirically observe consistent performance enhancement, as shown in Sec. 4 of our main paper.

2 Analysis of Meta-Optimization Rule

Inspired by [13, 8, 21], we here analyze the meta-optimization rule. The overall objective of our meta-optimization rule is formulated as:

$$\begin{aligned} &\min_{\phi} \ \big\{L_{v\_tr}(\phi) + L_{v\_te}(\phi')\big\} \\ =&\min_{\phi} \ \big\{L_{v\_tr}(\phi) + L_{v\_te}\big(\phi - \alpha \nabla L_{v\_tr}(\phi)\big)\big\} \end{aligned} \tag{1}$$

And the first-order Taylor expansion of a function can be formulated as:

$$f(x) \approx f(b) + \big(\nabla f(b)\big) \cdot (x - b) \tag{2}$$

where b is an arbitrary point close to x. Hence, by regarding $\phi - \alpha \nabla L_{v\_tr}(\phi)$ as x and setting $b = \phi$, we can write the second term $L_{v\_te}\big(\phi - \alpha \nabla L_{v\_tr}(\phi)\big)$ in our overall objective approximately as its first-order Taylor expansion . The overall objective can then be approximately reformulated as:

$$\begin{aligned} &\min_{\phi} \ \big\{L_{v\_tr}(\phi) + L_{v\_te}\big(\phi - \alpha \nabla L_{v\_tr}(\phi)\big)\big\} \\ \approx&\min_{\phi} \ \big\{L_{v\_tr}(\phi) + L_{v\_te}(\phi) - \alpha\big(\nabla L_{v\_tr}(\phi)\big) \cdot \big(\nabla L_{v\_te}(\phi)\big)\big\} \end{aligned} \tag{3}$$

This means that the overall objective of our meta-optimization is approximately equivalent to simultaneously (1) minimize the loss value on both the virtual training set D_{v_tr} and the virtual testing set D_{v_te} (i.e. $\min_{\phi} \{L_{v\_tr}(\phi) + L_{v\_te}(\phi)\}$); (2) maximize the term $(\nabla L_{v\_tr}(\phi)) \cdot (\nabla L_{v\_te}(\phi))$, in which (1) optimizes the confidence estimator to perform well on the virtual training and testing sets same as the objective of conventional joint-learning scheme and (2) maximizes the dot product of the gradient of $L_{v\_tr}(\phi)$ and the gradient of $L_{v\_te}(\phi)$.

Specifically, as shown by [19], maximizing the dot product of two gradients encourages the directions of the gradients to be similar. Hence, (2) encourages the confidence estimator to optimize on the virtual training set D_{v_tr} and the virtual testing set D_{v_te} to a similar direction, and hence learn invariant knowledge across the virtual training and testing sets. Moreover, as we intentionally design the virtual training and testing sets to have different distributions between them in our framework, by driving the confidence estimator to learn invariant knowledge across different distributions (i.e. incorporating (2) in our objective), the confidence estimator is guided to learn more *distribution-generalizable* knowledge.

Our experiments also show the efficacy of our meta-learning scheme, as shown in Tab. 5 of our main paper.

3 Details of Set Construction for Data Input *I*

During set construction for data input I, as discussed in Sec. 3.2 in the main paper, we need to first calculate the mean and the standard deviation of each convolutional layer's feature map of each input image i from D^I. Here we use i^l to denote the feature map from the l layer of the input image i. Then to calculate its mean $\mu(i^l)$ and standard deviation $\sigma(i^l)$, following [15], we first calculate the mean $\mu_c(i^l)$ and the standard deviation $\sigma_c(i^l)$ of each channel of the feature map i^l as:

$$\mu_c(i^l) = \frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} i_c^l \tag{4}$$

$$\sigma_c(i^l) = \sqrt{\frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} (i_c^l - \mu_c(i^l))^2} \tag{5}$$

where $C \times H \times W$ is the shape of the feature map i^l. After that, $\mu(i^l)$ and $\sigma(i^l)$ are constructed respectively as the concatenation of the calculated $\mu_c(i^l)$ and $\sigma_c(i^l)$ across the C channels of i^l.

Then to build a convolutional feature statistics vector to represent each input image i from D^I, we use $i^1, i^2, ..., i^M$ to denote all the feature maps of the image i learned from all the M convolutional layers of the confidence estimator, and thus the convolutional feature statistics vector $V_{stats}(i)$ of the image $i \in D^I$ is constructed as:

$$V_{stats}(i) = \left\{\mu(i^1), \sigma(i^1), ..., \mu(i^M), \sigma(i^M)\right\} \tag{6}$$

In this way, we can get the convolutional feature statistics vectors for all the images in D^I. After that, we cluster D^I into N clusters by applying the K-means algorithm on the images based on their convolutional feature statistics vectors, and finally, based on the clustering results, we can construct the virtual training and testing sets for data input I at the start of each even-numbered iteration, as shown in our main paper.

4 Visualization of Set Construction for Data Input *I*

To further demonstrate the effectiveness of our set construction method for data input I, we present some visualizations on how some clusters look like after we cluster them following our set construction method. As data distribution shifts exist within the same dataset [15, 17], to guide the confidence estimator to learn more *distribution generalizable* knowledge, we expect the clusters we construct to express such data distribution shifts. As shown from Fig. 1 to Fig. 4, the clusters we construct contain distinct style and domain characteristics and express the data distribution shifts, which demonstrates the general effectiveness of our set construction method for data input I across different datasets we use.

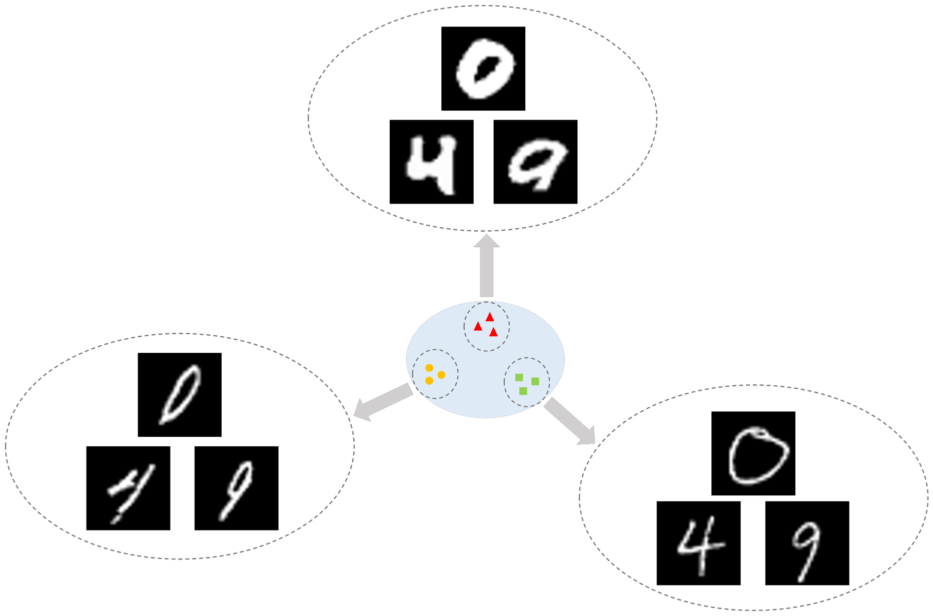


Fig. 1. Visualization of clusters we construct on MNIST training set [11].

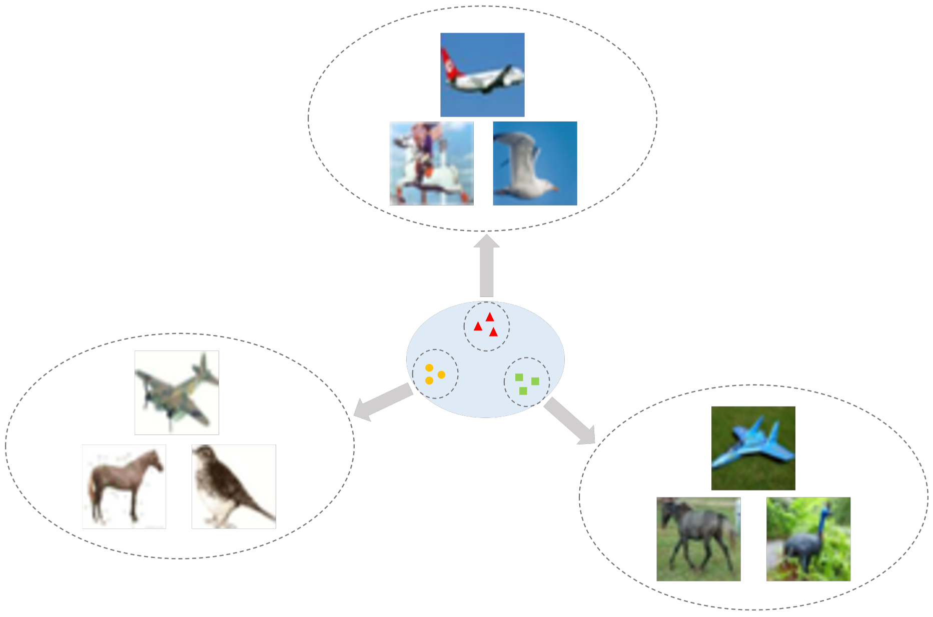


Fig. 2. Visualization of clusters we construct on CIFAR-10 training set [10].

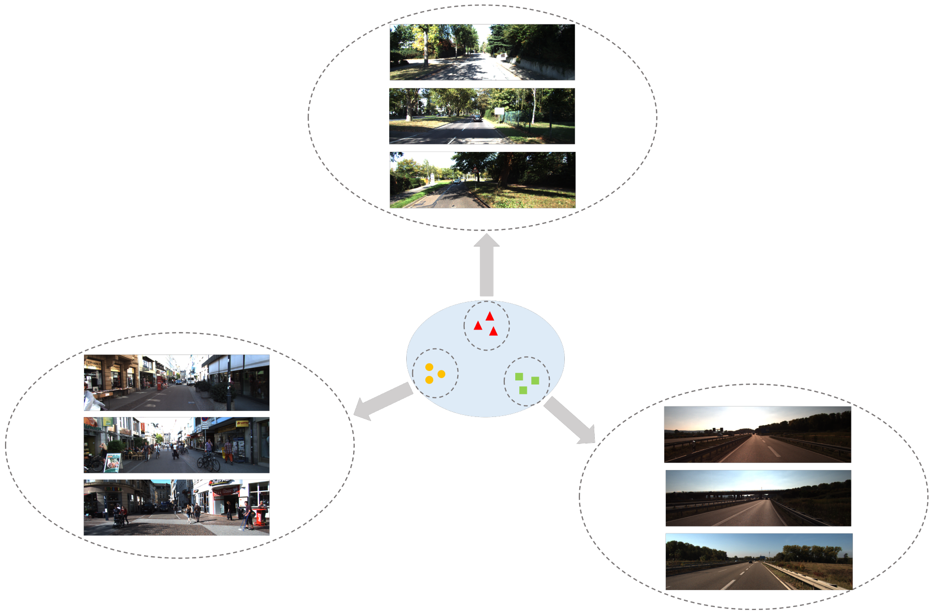


Fig. 3. Visualization of clusters we construct on KITTI Eigen-split training set [4, 7, 20].

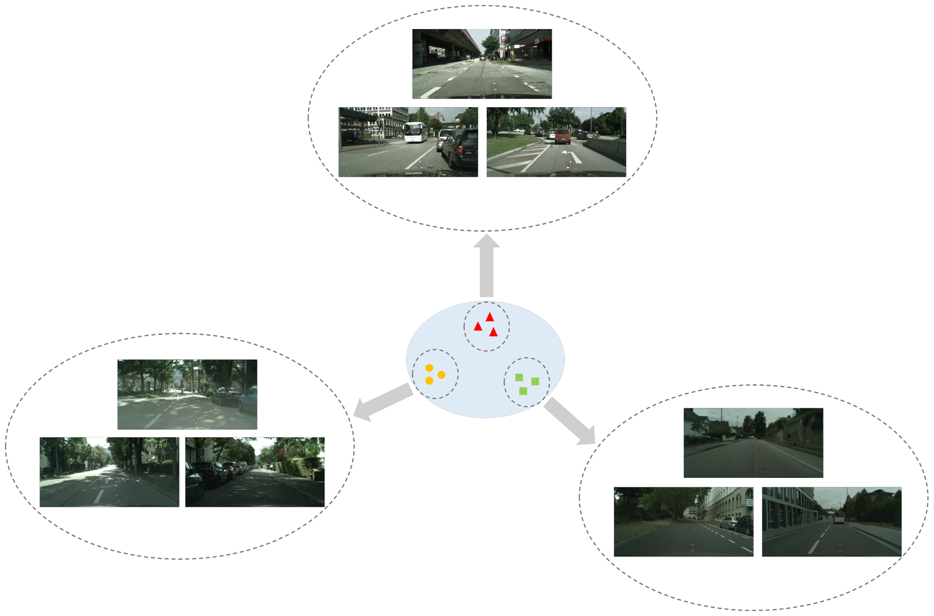


Fig. 4. Visualization of clusters we construct on Cityscapes training set [3].

5 Implementation Details of Representative methods

We reimplement various representative methods to tackle label imbalance and improve out-of-distribution data generalization in the main paper to compare them with our framework. Here we explain how we reimplement them more in detail. (1) **Reweight:** To tackle label imbalance, we assign the inverted sample fractions respectively to loss terms w.r.t. correct and incorrect task model predictions as weights. (2) **Resample [1]:** To tackle label imbalance, we up-sample the data with incorrect task model predictions to balance with the data with correct task model predictions during training. (3) **Dropout [6]:** Since dropout can be used to improve out-of-distribution data generalization [5], we also reimplement dropout by adding dropout layers into the network architecture of the confidence estimator and setting the dropout probability $p = 0.4$. (4) **Focal loss [14]:** Since focal loss can be used to both tackle label imbalance [12] and improve out-of-distribution data generalization [16], we use focal loss as the loss function of our confidence estimator and follow its default hyperparameters ($\gamma = 2.0$, $\alpha = 2.5$) in [14]. (5) **Mixup [23]:** To improve out-of-distribution data generalization, we also combine pairs of input samples and use these combinations together with original input samples during training following [23].

6 Additional Ablation Studies

Impact of splitting D^C into different subset proportions. In our experiments, we randomly select 60% of data from D^C to construct the first subset $D^{C,1}$, and use the remaining 40% as the second subset $D^{C,2}$. Here we assess two other variants, one where 50% of data is in the first subset (50% **for first subset**), and one where 70% of data is in the first subset (70% **for first subset**). Note that the remaining data (50% and 30% respectively) is used as the second subset $D^{C,2}$. As shown in Tab. 1, our framework and these two variants all achieve a better performance than the baseline of our framework (SLURP), which demonstrates the robustness of our framework to this hyperparameter.

Table 1. Ablation studies conducted on the use of different subset proportions when splitting D^C.

Method	CityScapes [3]			CityScapes Foggy s = 3 [18]			CityScapes Rainy s = 3 [9]		
	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑
Baseline(SLURP)	3.05	6.55	0.849	3.41	5.05	0.801	3.08	5.80	0.857
50% for first subset	0.71	0.74	0.925	0.99	0.61	0.935	1.14	0.86	0.907
60% for first subset	0.60	0.62	0.933	0.93	0.58	0.938	1.08	0.80	0.909
70% for first subset	0.69	0.70	0.929	1.01	0.64	0.932	1.17	0.89	0.904

Impact of using different numbers of clusters N. During set construction, we cluster D^I into N clusters, where N is set to 6 in our experiments. We evaluate other choices of N in Tab. 2. All variants ($N = 2$, $N = 4$, $N = 6$ and $N = 8$) outperform the baseline method (i.e., SLURP [22]) by a large margin, demonstrating that our framework is fairly robust and insensitive to the choice of N, and does not require intensive tuning of this hyperparameter.

Table 2. Ablation studies conducted on the number of clusters N.

Method	CityScapes [3]			CityScapes Foggy s = 3 [18]			CityScapes Rainy s = 3 [9]		
	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑
Baseline(SLURP)	3.05	6.55	0.849	3.41	5.05	0.801	3.08	5.80	0.857
2 clusters	1.03	1.30	0.911	1.17	1.04	0.911	1.24	1.86	0.902
4 clusters	0.63	0.70	0.931	0.94	0.68	0.936	1.16	1.18	0.906
6 clusters	0.60	0.62	0.933	0.93	0.58	0.938	1.08	0.80	0.909
8 clusters	0.60	0.63	0.933	0.94	0.58	0.938	1.07	0.81	0.910

Evaluation of different weights of reweight. In our experiments in the main paper, we reimplement reweight method by assigning the inverted sample fractions respectively to loss terms w.r.t. correct and incorrect task model predictions as weights. Here, we also evaluate other different weights of reweight method and compare them with our framework. Specifically, denote w^+ as weight assigned to loss term w.r.t. correct task model predictions and w^- as weight assigned to loss term w.r.t. incorrect task model predictions, we evaluate four other weights including (w^+, w^-) = (1,5), (w^+, w^-) = (1,10), (w^+, w^-) = (1,20), and (w^+, w^-) = (1,40). As shown in Tab. 3, our framework (tackling label imbalance only) outperforms all these different variants of the reweight method, which further demonstrates the effectiveness of our framework.

Table 3. Evaluation of different weights of reweight. w^+ and w^- respectively denote weights assigned to the loss terms w.r.t. correct and incorrect task model predictions.

Method (w^+, w^-)		CityScapes [3]			CityScapes Foggy s = 3 [18]			CityScapes Rainy s = 3 [9]		
		AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑
Baseline(SLURP)		3.05	6.55	0.849	3.41	5.05	0.801	3.08	5.80	0.857
SLURP + Reweight	(1,5)	2.89	6.02	0.852	3.29	4.97	0.804	2.96	5.67	0.859
SLURP + Reweight	(1,10)	2.77	5.89	0.855	3.20	4.78	0.809	2.72	5.42	0.862
SLURP + Reweight	(1,20)	2.59	5.48	0.860	2.98	4.49	0.817	2.51	5.20	0.863
SLURP + Reweight	(1,40)	2.65	5.66	0.859	3.04	4.55	0.814	2.59	5.26	0.863
SLURP + Ours(*tackling label imbalance only*)		1.33	1.76	0.908	1.98	2.34	0.864	1.71	2.55	0.889

Impact of constructing distributionally different virtual training and testing sets. In our experiments, we construct virtual training and testing sets to have different distributions between them in each iteration of our virtual training and testing scheme (**Distributionally Different Set Construction**). Here we assess the variant to randomly select samples to construct virtual training and testing sets in each iteration (**Random Set Construction**). As shown in Tab. 4, our framework achieves better performance than this variant, which demonstrates the effectiveness of constructing virtual training and testing sets to have different distributions.

Table 4. Ablation studies conducted on the effectiveness of constructing virtual training and testing sets to have different distributions.

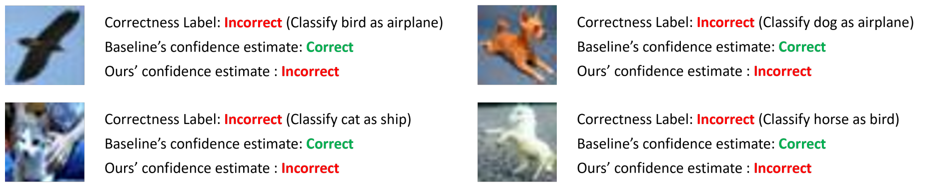
Method	CityScapes [3]			CityScapes Foggy s = 3 [18]			CityScapes Rainy s = 3 [9]		
	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑
Baseline(SLURP)	3.05	6.55	0.849	3.41	5.05	0.801	3.08	5.80	0.857
Random Set Construction	2.88	6.12	0.852	3.23	4.85	0.803	3.01	5.70	0.859
Distributionally Different Set Construction	0.60	0.62	0.933	0.93	0.58	0.938	1.08	0.80	0.909

7 Qualitative Results

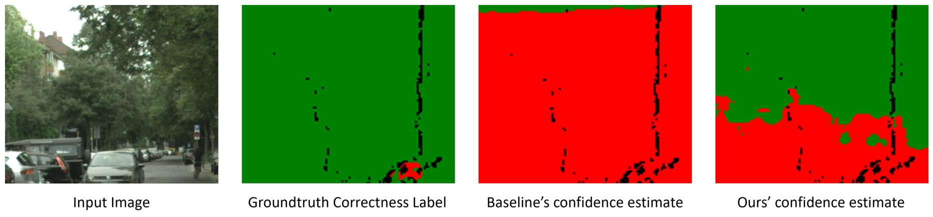
We present some qualitative results in Fig. 5 w.r.t. both the data with incorrect task model predictions and the out-of-distribution data inputs. As shown, compared with the baseline methods [2, 22], our framework performs better both under label imbalance and on out-of-distribution data inputs.

With respect to performing better under label imbalance, as the task model we use has made $> 93\%$ classifications correctly on CIFAR-10 [10], there exist many more correct task model predictions than incorrect ones. which leads to label imbalance for the confidence estimation on CIFAR-10 [10]. Hence, the confidence estimator is likely to be overly confident for incorrect task model predictions, which is undesirable. Hence, the framework we proposed aims to train the confidence estimator to perform better under label imbalance. As shown in Fig. 5(a), TCP [2] as the baseline method overconfidently estimates many images with incorrect task model predictions as correct, while our framework successfully identifies these incorrect predictions made by the task model and achieves a better performance under label imbalance.

With respect to handling the out-of-distribution data inputs, data distribution shifts occur within the same dataset, and more severe between different datasets, e.g. between the training data from existing datasets and testing data received during deployment under real-world conditions. Hence, in order for the confidence estimator to perform well in practical applications, it is important for the confidence estimator to perform well on out-of-distribution data inputs. As shown in Fig. 5(b), when an input image from the Cityscapes testing set [3] is passed to the confidence estimator trained on the KITTI Eigen-split training set [4, 7, 20] (i.e. an out-of-distribution data input from a different dataset), Our framework achieves much better performance compared to SLURP [22] as the baseline method.


Correctness Label: Incorrect (Classify bird as airplane)
Baseline's confidence estimate: Correct
Ours' confidence estimate : Incorrect
Correctness Label: Incorrect (Classify dog as airplane)
Baseline's confidence estimate: Correct
Ours' confidence estimate : Incorrect
Correctness Label: Incorrect (Classify cat as ship)
Baseline's confidence estimate: Correct
Ours' confidence estimate : Incorrect
Correctness Label: Incorrect (Classify horse as bird)
Baseline's confidence estimate: Correct
Ours' confidence estimate : Incorrect

(a) Results w.r.t the data with incorrect task model predictions from CIFAR-10 testing set [10] with TCP [2] as the baseline.


Input Image
Groundtruth Correctness Label
Baseline's confidence estimate
Ours' confidence estimate

(b) Results w.r.t an out-of-distribution data input from the Cityscapes testing set [3] evaluated using the confidence estimator trained on the KITTI Eigen-split training set [4, 7, 20]. Note that in the last three images, we use green-colored area to represent area that the task model correctly estimate the depth value, red-colored area to represent area that the task model incorrectly estimate the depth value, and *black-colored* area to represent area *without valid task model ground truth*.

Fig. 5. Qualitative results of our framework and the baseline methods [22, 2]. As shown, our framework demonstrates better performance both on data inputs with incorrect task model predictions and out-of-distribution data inputs.

8 Details of our used backbones

We use the same backbone as SLURP [22] for confidence estimation on monocular depth estimation and the same backbone as TCP [2] for confidence estimation on image classification. Here we introduce how these two backbones work respectively.

For confidence estimation on monocular depth estimation, as shown in Fig. 6, the backbone of SLURP passes both the data input I and the prediction map P through a feature encoder to output encoded features, and then further passes the fused encoded features through a context block to output the confidence estimate S. During training, the confidence estimator is optimized through a binary cross entropy loss function.

For confidence estimation on image classification, as shown in Fig. 7, the backbone of TCP passes only the data input I through a feature encoder to output encoded features, and then passes the encoded features through confidnet to output the confidence estimate S. During training, the confidence estimator is optimized through a TCP loss function. Note that compared to backbone of SLUPR, the backbone of TCP utilizes prediction P in the TCP loss function instead of through a feature encoder.

Note that our framework only modifies the training procedure of the confidence estimator, so we can directly apply our framework to the above two backbones with no adaptation to network structures although they have different network structures and use different loss functions.

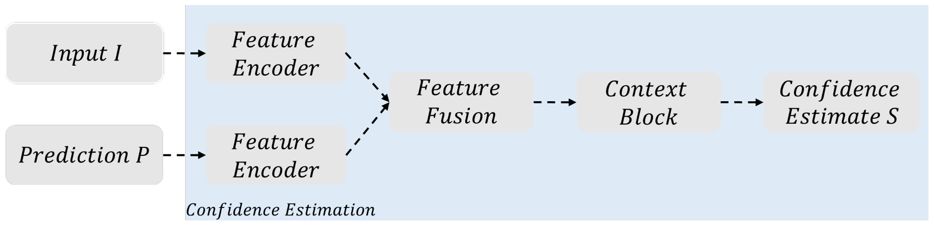


Fig. 6. Illustration of the backbone of SLURP [22]

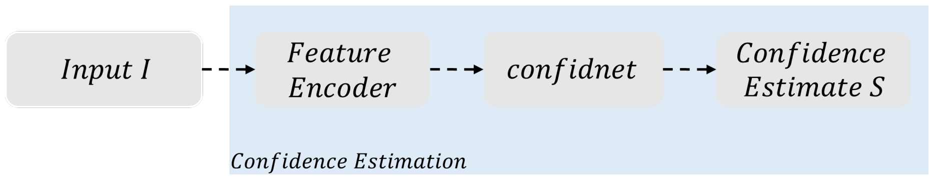


Fig. 7. Illustration of the backbone of TCP [2]

References

1. Burnaev, E., Erofeev, P., Papanov, A.: Influence of resampling on accuracy of imbalanced classification. In: Eighth international conference on machine vision (ICMV 2015). vol. 9875, pp. 423–427. SPIE (2015)
2. Corbière, C., Thome, N., Bar-Hen, A., Cord, M., Pérez, P.: Addressing failure prediction by learning model confidence. Advances in Neural Information Processing Systems **32** (2019)
3. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3213–3223 (2016)
4. Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. Advances in neural information processing systems **27** (2014)
5. Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B.: Sharpness-aware minimization for efficiently improving generalization. In: International Conference on Learning Representations (2021), `https://openreview.net/forum?id=6Tm1mposlrM`
6. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: international conference on machine learning. pp. 1050–1059. PMLR (2016)
7. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. International Journal of Robotics Research (IJRR) (2013)
8. Guo, J., Zhu, X., Zhao, C., Cao, D., Lei, Z., Li, S.Z.: Learning meta face recognition in unseen domains. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6163–6172 (2020)
9. Hu, X., Fu, C.W., Zhu, L., Heng, P.A.: Depth-attentional features for single-image rain removal. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8022–8031 (2019)
10. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
11. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
12. Li, B., Zheng, Z., Zhang, C.: Identifying incorrect classifications with balanced uncertainty. arXiv preprint arXiv:2110.08030 (2021)
13. Li, D., Yang, Y., Song, Y.Z., Hospedales, T.M.: Learning to generalize: Meta-learning for domain generalization. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
14. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017)
15. Matsuura, T., Harada, T.: Domain generalization using a mixture of multiple latent domains. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 11749–11756 (2020)
16. Mukhoti, J., Kulharia, V., Sanyal, A., Golodetz, S., Torr, P., Dokania, P.: Calibrating deep neural networks using focal loss. Advances in Neural Information Processing Systems **33**, 15288–15299 (2020)
17. Rabanser, S., Günnemann, S., Lipton, Z.: Failing loudly: An empirical study of methods for detecting dataset shift. Advances in Neural Information Processing Systems **32** (2019)

18. Sakaridis, C., Dai, D., Hecker, S., Van Gool, L.: Model adaptation with synthetic and real data for semantic dense foggy scene understanding. In: Proceedings of the european conference on computer vision (ECCV). pp. 687–704 (2018)
19. Shi, Y., Seely, J., Torr, P., N, S., Hannun, A., Usunier, N., Synnaeve, G.: Gradient matching for domain generalization. In: International Conference on Learning Representations (2022), `https://openreview.net/forum?id=vDwBW49HmO`
20. Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., Geiger, A.: Sparsity invariant cnns. In: 2017 international conference on 3D Vision (3DV). pp. 11–20. IEEE (2017)
21. Xu, L., Qu, H., Kuen, J., Gu, J., Liu, J.: Meta spatio-temporal debiasing for video scene graph generation. In: Proceedings of the European Conference on Computer Vision (ECCV) (2022)
22. Yu, X., Franchi, G., Aldea, E.: Slurp: Side learning uncertainty for regression problems. In: 32nd British Machine Vision Conference, BMVC 2021, Virtual Event / November 22-25, 2021 (2021)
23. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: International Conference on Learning Representations (2018)