Improving the Reliability for Confidence Estimation

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

Abstract. Confidence estimation, a task that aims to evaluate the trustworthiness of the model's prediction output during deployment, has received lots of research attention recently, due to its importance for the safe deployment of deep models. Previous works have outlined two important qualities that a reliable confidence estimation model should possess, i.e., the ability to perform well under label imbalance and the ability to handle various out-of-distribution data inputs. In this work, we propose a meta-learning framework that can simultaneously improve upon both qualities in a confidence estimation model. Specifically, we first construct virtual training and testing sets with some intentionally designed distribution differences between them. Our framework then uses the constructed sets to train the confidence estimation model through a *virtual training and testing* scheme leading it to learn knowledge that generalizes to diverse distributions. We show the effectiveness of our framework on both monocular depth estimation and image classification.

Keywords: Confidence estimation, Meta-learning.

1 Introduction

With the continuous development of deep learning techniques, deep models are becoming increasingly accurate on various computer vision tasks, such as image classification [22] and monocular depth estimation [25]. However, even highly accurate models might still commit errors [1, 18, 10], and these errors can potentially lead to serious consequences, especially in safety-critical fields, such as nuclear power plant monitoring [30], disease diagnosis [39], and self-driving vehicles [40]. Due to the severe implications of errors in these applications, it is crucial for us to be able to assess whether we can place confidence in the model predictions, before acting according to them. Hence, the task of confidence estimation (also known as trustworthiness prediction), which aims to evaluate the

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

confidence of the model's prediction during deployment, has received a lot of research attention recently [19, 5, 32].

Specifically, in confidence estimation, we would like to compute the confidence estimate $S \in \{0, 1\}$ for a prediction P made by a model regarding input I, where S estimates if prediction P is correct (1) or not (0). In this paper, for clarity, the *task model* refers to the deep model that produces predictions P on the main task; confidence estimation for P is performed by a separate confidence estimation model, which we refer to as the *confidence estimator*, as shown in Fig. 1. Many previous works [5, 32, 26, 47] have proposed to train such a confidence estimator to conduct confidence estimation more reliably.

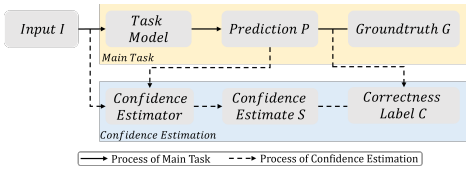

Fig. 1. Illustration of confidence estimation.

Some recent works [32, 26] have noted that a reliable confidence estimator should *perform well under label imbalance*. This is because confidence estimators use the *correctness of task model predictions (C) as labels*, which are often imbalanced. As shown in Fig. 1, correctness labels C are produced by checking for consistency between predictions P and ground truths G, where $C = 1$ if P is correct and $C = 0$ otherwise. Thus, since many task models have achieved good performance on computer vision tasks (e.g., Small ConvNet [19] achieves $> 99\%$ for MNIST [24] and VGG-16 [41, 31] achieves $> 93\%$ for CIFAR-10 [21] in image classification), there are often many more correct predictions (where $C = 1$) than incorrect ones (where $C = 0$), which leads to label imbalance for the confidence estimation task. If this label imbalance is not accounted for during training, the confidence estimator is likely to be overly confident [32, 26] for incorrect predictions (where $C = 0$), which is undesirable.

On the other hand, some other works [35, 43] have suggested that the ability to handle *out-of-distribution data inputs* (I) is important for confidence estimation. Out-of-distribution data occurs due to distribution shifts in the data – such data distribution shifts can occur within the same dataset [33], but are generally more severe between different datasets, e.g. between the training data from existing datasets and testing data received during deployment under real-world conditions. If the confidence estimator does not learn to handle out-of-distribution inputs, it will tend to perform badly whenever an out-of-distribution input sample I is fed into the task model, which affects its utility in practical applications.

In this paper, we aim to improve the reliability of our confidence estimator in terms of both the above-mentioned qualities; we improve its ability to tackle label imbalance of C, as well as to handle various out-of-distribution inputs I. Specifically, we observe that these qualities actually share a common point – they are acquired when the confidence estimator learns to *generalize to diverse distributions*. If a confidence estimator learns knowledge that can generalize to diverse distributions, it will be able to tackle diverse correctness label (C) distributions, which includes distributions where $C = 0$ is more common, and can

thus better tackle the imbalanced label problem; it will also be able to tackle diverse input (I) distributions, which improves performance on out-of-distribution data. Based on this novel perspective, we propose to improve upon both of these qualities *simultaneously through a unified framework*, that allows the confidence estimator to learn to generalize, and perform well on distributions that might be different from the distributions (of both C and I) seen during training. In order to achieve this, we incorporate *meta-learning* into our framework.

Meta-learning, also known as "*learning to learn*", allows us to train a model that can *generalize well to different distributions*. Specifically, in some meta-learning works [9, 28, 13, 2, 16, 46], a *virtual testing set* is used to mimic the testing conditions during training, so that even though training is mainly done on a *virtual training set* consisting of training data, performance on the testing scenario is improved. In our work, we construct our virtual testing sets such that they simulate various distributions that are different from the virtual training set, which will push our model to learn *distribution-generalizable knowledge to perform well on diverse distributions*, instead of learning *distribution-specific knowledge that only performs well on the training distribution*. In particular, for our confidence estimator to learn distribution-generalizable knowledge and tackle diverse distributions of C and I, we intentionally construct virtual training and testing sets that simulate the different distribution shifts of C and I, and use them for meta-learning.

The contributions of our work are summarized as follows. 1) We propose a novel framework, which incorporates meta-learning to learn a *confidence estimator* to produce confidence estimates more reliably. 2) By carefully constructing virtual training and testing sets that simulate the training and various testing scenarios, our framework can learn to generalize well to different correctness label distributions and input distributions. 3) We apply our framework upon state-of-the-art confidence estimation methods [5, 47] across various computer vision tasks, including image classification and monocular depth estimation, and achieve consistent performance enhancement throughout.

2 Related Work

Confidence Estimation. Being an important task that helps determine whether a deep predictor's predictions can be trusted, confidence estimation has been studied extensively across various computer vision tasks [14, 11, 19, 5, 34, 36, 32, 44, 4, 35, 26, 43, 47]. At the beginning, Hendrycks and Gimpel [14] proposed Maximum Class Probability utilizing the classifier softmax distribution, Gal and Ghahramani [11] proposed MCDropout from the perspective of uncertainty estimation, and Jiang et al. [19] proposed Trust Score to calculate the agreement between the classifier and a modified nearest-neighbor classifier in the testing set. More recently, the idea of *separate confidence estimator* was introduced by several works [5, 47]. Specifically, these works proposed to fix the task model, and instead conduct confidence estimation via a separate confidence estimator. Notably, Corbiere et al. [5] proposed a separate confidence estimator called Con-

fidnet and a new loss function called True Class Probability. Subsequently, Yu et al. [47] proposed SLURP, a generic confidence estimator for regression tasks, that is specially targeted at task models that perform monocular depth estimation.

In this paper, we also build a separate confidence estimator, since it has the benefit of not affecting the main task performance. Different from previous works, we propose a novel meta-learning framework that simultaneously improves the performance of the confidence estimator under label imbalance and on out-of-distribution input data, in a unified manner.

Label Imbalance in Confidence Estimation. Recently, using the the correctness of task model predictions (C) as labels, many existing confidence estimation methods [14, 11, 5] have been shown to suffer from the label imbalance problem. To solve this problem and enable the *confidence estimator* to perform well under label imbalance, various methods have been proposed. Luo et al. [32] proposed a loss function called Steep Slope Loss to separate features w.r.t. correct and incorrect task model predictions from each other. Afterwards, Li et al. [26] proposed an extension to True Class Probability [5] that uses a Distributional Focal Loss to focus more on predictions with higher uncertainty. Unlike previous methods that design strategies to handle a specific imbalanced distribution of correct and incorrect labels, we adopt a novel perspective, and tackle the label imbalance problem through meta-learning, which allows our confidence estimator to learn *distribution-generalizable knowledge to tackle a variety of diverse label distributions*. This is done through construction of virtual testing sets that *simulate various different label distributions*.

Confidence Estimation on Out-of-distribution Data. As various distribution shifts exist between the training and testing data in real-world applications, the handling of out-of-distribution data inputs (I) is important for reliable confidence estimation. To this end, Mukhoti et al. [35] proposed to replace the cross entropy loss with the focal loss [29], and utilize its implicit regularization effects to handle out-of-distribution data. Tomani et al. [43] proposed to handle out-of-distribution data via applying perturbations on data from the validation set. However, as these previous methods either emphasizes on the rare samples or fine-tunes on an additional set of samples, they can still be prone to overfit these rare samples or the additional set of samples. Differently, in this work, we propose to use meta-learning and optimize the model through feedbacks from diverse virtual sets with diverse distributions. Thus, we can enable our model to learn knowledge that is more generalizable to various out-of-distribution data.

Meta-learning. MAML [9], a popular meta-learning method, was originally designed to learn a good weight initialization that can quickly adapt to new tasks in testing, which showed promise in few-shot learning. Subsequently, its extension [28], which requires no model updating on the unseen testing scenarios, has been applied beyond few-shot learning, to enhance model performance [13, 2, 16, 46]. Differently, we propose a novel framework via meta-learning to perform *more reliable confidence estimation*. Through performing meta-learning on carefully constructed virtual training and virtual testing sets, we simultaneously improve

the ability of our confidence estimator to generalize well to different distributions of C and I.

3 Method

To conduct confidence estimation reliably, previous works have suggested two important qualities that a model should possess: the ability to *perform well under label imbalance*, and the ability to *handle various out-of-distribution data inputs*. We find that both qualities are actually acquired when the confidence estimator is trained to perform well across different distributions (w.r.t. either the correctness label C or the data input I). Hence, to train a more reliable confidence estimator, we leverage upon meta-learning that allows our confidence estimator to learn more distribution-generalizable knowledge to better tackle diverse distributions – which is achieved by obtaining feedback from a *virtual testing set* while concurrently updating using a *virtual training set*. A crucial part of our meta-learning algorithm is the virtual testing set construction, which needs to simulate diverse distributions to provide good feedback to the confidence estimator. Specifically, at the start of each iteration, from the training set D, we first construct a virtual training set D_{v_tr} and a virtual testing set D_{v_te}, such that there are *intentionally designed distribution differences between them*. To optimize the confidence estimator to possess both qualities discussed above, the virtual training set D_{v_tr} and the virtual testing set D_{v_te} are constructed to have different distributions of correctness labels C every odd-numbered iteration and different distributions of data inputs I every even-numbered iteration. After constructing D_{v_tr} and D_{v_te}, our framework then uses them to train the confidence estimator through a *virtual training and testing* procedure based on meta-learning.

Below, we first describe the *virtual training and testing* scheme we use to train the confidence estimator in Sec. 3.1. Next, in Sec. 3.2, we discuss how we construct our virtual training and virtual testing sets at the start of each iteration. Finally, we describe our framework as a whole in Sec. 3.3.

3.1 Virtual Training and Testing

As mentioned above, at the start of each iteration, we first construct a virtual training set D_{v_tr} and a virtual testing set D_{v_te}, such that there are intentionally designed distribution differences between them. In this section, we assume that D_{v_tr} and D_{v_te} have been constructed, and describe how we utilize them via the *virtual training and testing* scheme to train the confidence estimator to generalize to different distributions.

Specifically, each iteration of the *virtual training and testing* scheme contains three steps: (1) **Virtual training**. We first virtually train the confidence estimator using the virtual training set D_{v_tr} to simulate the conventional training procedure of the confidence estimator. (2) **Virtual testing**. After that, the confidence estimator is assessed (i.e., virtually tested) on the virtual testing set

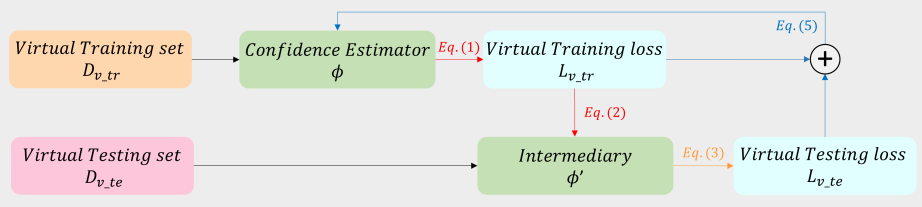

Fig. 2. Illustration of our virtual training and testing scheme. (1) In the virtual training step, we conduct updates on the confidence estimator parameters ϕ with the virtual training set D_{v_tr}, and obtain an intermediary ϕ' (which is indicated with red arrows). (2) The intermediary ϕ' is then evaluated on the virtual testing set D_{v_te} with a different distribution from D_{v_tr} to obtain the virtual testing loss L_{v_te} (which is indicated with the yellow arrow). (3) Lastly, the virtual training loss L_{v_tr} and the virtual testing loss L_{v_te} are used to update confidence estimator ϕ (indicated with blue arrows), such that it can generalize over diverse distributions and become more reliable.

D_{v_te}, which evaluates the performance on a different distribution from the virtual training set D_{v_tr}. (3) **Meta Optimization (Actual update)**. Finally, we incorporate the evaluation result (loss) calculated during virtual testing as a feedback to actually update the confidence estimator. This provides a feedback to the confidence estimator, that allows it to learn generalizable knowledge to tackle diverse distributions while training using the virtual training set D_{v_tr}. Below, we describe these three steps of the *virtual training and testing* scheme in more detail. We also demonstrate these three steps in Fig. 2.

Virtual Training. During virtual training, we simulate the conventional training procedure of the confidence estimator, and first train the confidence estimator via gradient descent with data from the virtual training set D_{v_tr}. Here we denote the confidence estimator parameters as ϕ, the learning rate for virtual training as α, and the loss function of the confidence estimator as L (e.g., binary cross entropy loss). We can calculate the virtual training loss L_{v_tr} as:

$$L_{v\_tr}(\phi) = L(\phi, D_{v\_tr}) \tag{1}$$

Using this loss, we can update our confidence estimator parameters ϕ via gradient descent:

$$\phi' = \phi - \alpha \nabla_{\phi} L_{v\_tr}(\phi) \tag{2}$$

Note that we do not actually update the confidence estimator to be ϕ' (hence the term "virtual"). Instead, the *virtually trained* ϕ' is just an intermediary to calculate L_{v_te} in next step, and simulates what training on D_{v_tr} would be like.

Virtual Testing. In this step, we evaluate how the virtually updated confidence estimator ϕ' (that is trained on D_{v_tr}) performs on the virtual testing set D_{v_te}, which has a different distribution to D_{v_tr}.

$$L_{v\_te}(\phi') = L(\phi', D_{v\_te}) \tag{3}$$

The computed virtual testing loss L_{v_te} measures the confidence estimator performance on D_{v_te}, after one simulated training step on D_{v_tr}, and can be used to provide feedback on how we can update the confidence estimator parameters such that it can better generalize to different distributions (as is done in next step).

Meta-optimization (Actual update). In the virtual training and virtual testing steps, we have computed the losses L_{v_tr} and L_{v_te} respectively. In this step, we use them to optimize our confidence estimator to perform well on diverse distributions, by obtaining feedback from L_{v_te} while concurrently updating using L_{v_tr}. We first formulate our overall objective as:

$$\begin{aligned} &\min_{\phi} \ \big\{L_{v\_tr}(\phi) + L_{v\_te}(\phi')\big\} \\ =&\min_{\phi} \ \big\{L_{v\_tr}(\phi) + L_{v\_te}\big(\phi - \alpha\nabla_{\phi} L_{v\_tr}(\phi)\big)\big\} \end{aligned} \tag{4}$$

We highlight that, in Eq. 4, our goal is to optimize ϕ, and ϕ' is just used as a helpful intermediary in calculating $L_{v\_te}(\phi')$. After constructing our overall objective, we can then update ϕ via gradient descent for meta-optimization as:

$$\phi \leftarrow \phi - \beta\nabla_{\phi}\Big(L_{v\_tr}(\phi) + L_{v\_te}\big(\phi - \alpha\nabla_{\phi} L_{v\_tr}(\phi)\big)\Big) \tag{5}$$

where β denotes the learning rate for meta-optimization. By updating the confidence estimator with the meta-optimization update rule in Eq. 5, the confidence estimator is updated with knowledge that is more *distribution-generalizable*, leading to a more reliable confidence estimator that is applicable to diverse distributions. We explain this in more detail below.

During virtual training, we first update the confidence estimator ϕ to an intermediary ϕ' in Eq. 2. In this step, the intermediary ϕ' can learn *distribution-specific knowledge* (that is only specifically applicable to the distribution of D_{v_tr}), as such knowledge can help improve performance on D_{v_tr}. On the other hand, we note that for the intermediary (trained on D_{v_tr}) to generalize well to the virtual testing set D_{v_te} in Eq. 3 (which has a *different distribution* compared to D_{v_tr} and where *distribution-specific knowledge from* D_{v_tr} *does not apply*), the intermediary needs to *avoid learning distribution-specific knowledge* when learning on D_{v_tr}, and instead *learn more distribution-generalizable knowledge*. This means that, the term $L_{v\_te}(\phi')$ in Eq. 3 provides a feedback which guides the learning towards acquiring more distribution-generalizable knowledge.

Importantly, based on our analysis above, as long as D_{v_tr} and D_{v_te} have *different distributions and cannot be tackled with the same distribution-specific knowledge*, the confidence estimator will be encouraged to avoid learning distribution-specific knowledge, and focus on learning more distribution-generalizable knowledge in the meta-optimization step. This also implies that, we do not aim to use the distribution of virtual testing set D_{v_te} to simulate the distribution of the real testing scenario (which is unknown during training) to learn distribution-generalizable knowledge that can tackle real testing scenarios. We also present a more theoretical analysis of the efficacy of the meta-optimization rule in the supplementary material.

3.2 Set Construction

In this section, we discuss how we construct a virtual training set D_{v_tr} and a virtual testing set D_{v_te} to have different distributions in each iteration of our virtual training and testing scheme (that is described in Sec. 3.1). Specifically, *at the start of each epoch*, we first split the training set D into two halves: D^C and D^I, which will be used to tackle the two different problems (w.r.t correctness labels C and data inputs I). Within the epoch, at the start of every odd-numbered iteration, we construct virtual training and testing sets from D^C to tackle the label imbalance problem; on the other hand, at the start of every even-numbered iteration, we construct virtual training and testing sets from D^I to tackle the out-of-distribution data input problem.

As there exist some differences between the distribution of C and the distribution of I (e.g., it is more difficult to characterize the distribution of data input I and find input distributions that are different), we propose different set construction methods for each of them that provide diverse testing distributions in practice. We highlight that, due to the *unified nature of our framework*, tackling of these two different problems have now been reduced to a more straightforward designing of their respective set construction methods. Below, we separately discuss each construction method.

Constructing sets for correctness label C. With respect to the correctness label C, we construct virtual training and testing sets with different distributions in two steps. **Step (C1)** *At the start of each epoch*, we first randomly split D^C into two subsets D_1^C and D_2^C, where the first subset D_1^C will be used to construct batches of D_{v_tr}, and the second subset D_2^C will be used to construct batches of D_{v_te}. Then, we pre-compute the correctness label C w.r.t. every sample in the second subset D_2^C to facilitate D_{v_te} construction in that epoch. **Step (C2)** *At the start of every odd-numbered iteration*, we randomly select a batch of data from the first subset D_1^C to construct a virtual training set D_{v_tr}. Next, we construct the virtual testing set D_{v_te} – we want its distribution of C to vary between iterations constantly, to simulate various distributions that are different from D_{v_tr}. Hence, we randomly select a percentage from 0 to 100% to set as the percentage of correct task model predictions (where $C = 1$) in D_{v_te} each time. Based on the sampled percentage of correct task model predictions, we randomly select a batch of samples from the second subset D_2^C to construct the virtual testing set D_{v_te} to approximately match that percentage. This way, our virtual testing set D_{v_te} will have a different distribution of C compared to D_{v_tr}, with a high probability.

Constructing sets for data input I. Besides, we also construct virtual training and testing sets to have different distributions w.r.t. the data input I. Note that, when using only a single dataset, constructing virtual training and testing sets to have different data input distributions is a difficult problem. Here we follow a simple and effective technique proposed in previous works [17, 33, 27] that can help to simulate a distribution shift within a dataset. Specifically, they found that the statistics (i.e., mean and standard deviation) computed spatially over the pixels within the convolutional feature map of an input image, are a

compact representation that effectively captures the style and domain characteristics of this image. Hence, we concatenate the convolutional feature statistics from all convolutional layers of our confidence estimator (into a single vector) as a representation of each input sample. Then, following [33], we use a K-means clustering technique to separate the convolutional feature statistics vectors of all the data in D^I into different clusters, such that a data distribution shift is simulated between clusters, that will be used for constructing virtual training and testing sets with different distributions.

Specifically, our set construction for data input I is done in two steps. **Step (I1)** *At the start of each epoch*, we first cluster D^I into N clusters by applying the K-means algorithm on the convolutional feature statistics vectors of samples in D^I. Among the N clusters, we randomly select one cluster as D^I_1 that will be used to construct D_{v_tr} in this epoch. **Step (I2)** *Then at the start of every even-numbered iteration*, we first randomly select a batch of data from the selected cluster D^I_1 to construct the virtual training set D_{v_tr}. After that, we randomly select a cluster from the remaining $N-1$ clusters, and select a batch of data from this cluster to construct the virtual testing set D_{v_te}. For more details, please refer to the Supplementary.

After constructing virtual training and testing sets as discussed above, during experiments, we empirically observe consistent performance enhancement, as shown in Sec. 4, which shows the effectiveness of our set construction method.

3.3 Overall Training and Testing Scheme

In the above two sections, we have described the *virtual training and testing* scheme and how we construct our virtual training and virtual testing sets. In this section, we summarize them and discuss the overall training and testing scheme of our framework. Specifically, in the training procedure of the confidence estimator, at the start of each iteration, we first construct the virtual training and testing sets to have different distributions (w.r.t correctness label C in odd-numbered iterations and data input I in even-numbered iterations) following Sec. 3.2. After that, the constructed virtual training and testing sets are used to train the confidence estimator through the *virtual training and testing* scheme as discussed in Sec. 3.1. Hence, we alternatingly deal with the label imbalance problem and the handling of out-of-distribution inputs over iterations, resulting a simulatenous tackling of both problems. We demonstrate the overall training scheme of our framework in Alg. 1. During testing, we follow the evaluation procedure of previous works [47, 5].

4 Experiments

In this section, in order to verify the effectiveness of our proposed framework, we conduct experiments on various different tasks including monocular depth estimation and image classification. For monocular depth estimation, we only modify the training procedure by adding our framework and follow all the other

Algorithm 1: Overall Training Scheme

Initialize ϕ.
for E *epochs* **do**
  Randomly split D into two halves: D^C and D^I.
  Process D^C and D^I following Step (C1) and Step (I1) in Sec. 3.2 respectively.
  for T *iterations* **do**
    if T *is odd* **then**
      Construct D_{v_tr} and D_{v_te} from D^C, following Step (C2) in Sec. 3.2.
    else
      Construct D_{v_tr} and D_{v_te} from D^I, following Step (I2) in Sec. 3.2.
    Calculate the virtual training loss L_{v_tr} on D_{v_tr} using Eq. 1: $L_{v\_tr}(\phi) = L(\phi, D_{v\_tr})$.
    Calculate an updated version of confidence estimator (ϕ') using Eq. 2: $\phi' = \phi - \alpha \nabla_\phi L_{v\_tr}(\phi)$.
    Calculate the virtual testing loss L_{v_te} on D_{v_te} using Eq. 3: $L_{v\_te}(\phi') = L(\phi', D_{v\_te})$.
    Update using Eq. 5: $\phi \leftarrow \phi - \beta \nabla_\phi \Big(L_{v\_tr}(\phi) + L_{v\_te}\big(\phi - \alpha \nabla_\phi L_{v\_tr}(\phi)\big) \Big)$.

experiment settings of [47] for evaluation on various testing scenarios. For image classification, similarly, we merely change the training procedure to include our framework, and follow all the other experiment settings of [5] to test our proposed method. We conduct all our experiments on an RTX 3090 GPU, and fix the task model during confidence estimator training.

4.1 Confidence estimation on Monocular Depth Estimation

Settings and Implementation Details. For monocular depth estimation, we follow [47] and conduct two groups of experiments to evaluate our proposed framework. In the first experiment, we train our confidence estimator on KITTI Eigen-split training set [8, 12, 45], and evaluate the trained confidence estimator on two testing scenarios: KITTI Eigen-split testing set from the *same dataset*, and Cityscapes [6] testing set from a *different dataset*.

Table 1. Experiment results of confidence estimation on monocular depth estimation, with our model **trained on KITTI Eigen-split training set** following the setting in [47]. Our method performs the best across all metrics.

Method	KITTI [12, 45, 8]			CityScapes [6]		
	AUSE-RMSE↓	AUSE-Abserl↓	AUROC↑	AUSE-RMSE↓	AUSE-Abserl↓	AUROC↑
MCDropout [11]	8.14	9.48	0.686	9.42	9.52	0.420
Empirical Ensembles	3.17	5.02	0.882	11.56	13.14	0.504
Single PU [20]	1.89	4.59	0.882	9.91	9.96	0.386
Deep Ensembles [23]	1.68	4.32	0.897	11.47	9.36	0.501
True Class Probability [5]	1.76	4.24	0.892	10.48	5.75	0.519
SLURP [47]	1.68	4.36	0.895	9.48	10.90	0.400
SLURP + Reweight	1.67	4.29	0.896	9.39	10.41	0.402
SLURP + Resample [3]	1.67	4.28	0.896	9.37	10.35	0.404
SLURP + Dropout [42]	1.67	4.20	0.896	9.29	10.01	0.412
SLURP + Focal loss [29]	1.67	4.18	0.895	9.30	10.14	0.410
SLURP + Mixup [48]	1.67	4.15	0.896	9.17	10.01	0.420
SLURP + Resampling + Mixup	1.67	4.07	0.896	8.99	9.64	0.431
SLURP + Ours(*tackling label imbalance only*)	1.66	3.84	0.897	8.75	7.79	0.509
SLURP + Ours(*tackling out-of-distribution inputs only*)	1.66	3.90	0.897	8.54	6.90	0.524
SLURP + Ours(full)	**1.65**	**3.62**	**0.898**	**8.26**	**5.32**	**0.601**

In the second experiment, we further evaluate our framework under *different weather conditions*. We fine tune our trained confidence estimator on Cityscapes training set, and evaluate it on several testing scenarios: Cityscapes testing set, Foggy Cityscapes-DBF [38] testing set with three severity levels, and Rainy Cityscapes [15] testing set with three severity levels.

Table 2. Experiment results of confidence estimation on monocular depth estimation, with our model **fine-tuned on CityScapes [6]** following the setting in [47]. In this table, s indicates severity. The higher s is, more severe the rain or the fog is. Our method performs the best across all metrics and testing scenarios.

Method	CityScapes [6]			CityScapes Foggy $s=1$ [38]			CityScapes Foggy $s=2$ [38]			CityScapes Foggy $s=3$ [38]			CityScapes Rainy $s=1$ [15]			CityScapes Rainy $s=2$ [15]			CityScapes Rainy $s=3$ [15]		
	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑
MCDropout [11]	7.72	8.13	0.705	7.06	8.73	0.659	7.14	8.36	0.667	7.30	8.27	0.665	7.80	8.36	0.700	7.82	8.20	0.704	7.84	7.87	0.715
Empirical Ensembles	8.20	7.50	0.786	7.29	6.92	0.757	6.90	6.48	0.767	6.66	6.03	0.778	7.82	7.33	0.783	7.53	7.09	0.791	7.28	6.80	0.801
Single PU [20]	4.35	6.44	0.741	4.17	6.55	0.731	4.27	6.79	0.731	4.35	6.44	0.742	3.42	6.78	0.842	3.42	6.55	0.847	3.48	6.19	0.851
Deep Ensembles [23]	3.03	6.81	0.856	3.42	6.68	0.746	3.35	6.24	0.756	3.28	5.85	0.767	3.05	6.58	0.852	2.98	6.35	0.857	2.93	6.01	0.863
True Class Probability [5]	4.05	6.34	0.821	4.89	7.26	0.697	4.68	6.86	0.714	4.59	6.64	0.729	3.98	6.21	0.824	3.86	6.02	0.833	3.70	5.78	0.846
SLURP [47]	3.05	6.55	0.849	3.39	5.62	0.788	3.36	5.28	0.794	3.41	5.05	0.801	3.04	6.25	0.847	3.01	6.06	0.852	3.08	5.80	0.857
SLURP + Reweight	2.56	5.47	0.861	2.71	5.14	0.804	2.89	5.06	0.811	2.93	4.46	0.819	2.85	6.07	0.854	2.89	5.87	0.868	2.45	5.14	0.864
SLURP + Resample [3]	2.51	5.32	0.865	2.69	5.11	0.805	2.76	4.99	0.813	2.89	4.27	0.823	2.72	5.99	0.857	2.77	5.84	0.871	2.31	4.85	0.866
SLURP + Dropout [42]	2.88	5.39	0.857	2.52	4.91	0.813	2.55	4.78	0.819	2.74	4.01	0.835	2.56	5.68	0.863	2.65	5.47	0.880	2.27	4.77	0.870
SLURP + Focal loss [29]	2.75	5.20	0.859	2.49	4.87	0.816	2.47	4.69	0.825	2.67	4.09	0.830	2.49	5.55	0.867	2.41	5.23	0.884	2.18	4.81	0.867
SLURP + Mixup [48]	2.49	5.13	0.866	2.28	4.59	0.827	2.33	4.41	0.830	2.44	3.92	0.847	2.41	5.44	0.869	2.34	5.11	0.889	2.03	4.51	0.872
SLURP + Resample + Mixup	2.31	4.97	0.869	2.05	4.29	0.836	2.08	4.24	0.842	2.29	3.79	0.853	2.33	5.14	0.880	2.09	4.98	0.877	1.94	3.77	0.877
SLURP + Ours(*tackling label imbalance only*)	1.33	1.76	0.908	1.84	2.64	0.874	1.90	2.51	0.871	1.98	2.34	0.864	1.81	3.17	0.889	1.77	3.10	0.891	1.71	2.55	0.889
SLURP + Ours(*tackling out-of-distribution inputs only*)	1.42	1.88	0.900	1.61	2.19	0.890	1.66	2.20	0.891	1.76	2.09	0.881	1.59	2.78	0.901	1.60	2.75	0.904	1.63	1.97	0.890
SLURP + Ours(full)	**0.60**	**0.62**	**0.933**	**0.73**	**0.63**	**0.934**	**0.80**	**0.58**	**0.937**	**0.93**	**0.58**	**0.938**	**0.85**	**0.69**	**0.923**	**0.96**	**0.68**	**0.925**	**1.08**	**0.80**	**0.909**

We emphasize that in these experiments, the distribution of I will face a large shift from training conditions due to the cross-dataset/cross-weather setting. Moreover, there is also obvious imbalance in the distribution of C. Specifically, **the distribution of correct and incorrect labels of C in the KITTI Eigen-split training set is quite imbalanced (99.8%:0.2%)**. This means, there are both obvious label (C) imbalance problem and input (I) out-of-distribution problem.

In both above-mentioned experiments, we use the same backbone as SLURP [47], which is described in more detail in the supplementary material. Following the setting in [47], we use the area under sparsification error corresponding to square error (**AUSE-RMSE**), the area under sparsification error corresponding to absolute relative error (**AUSE-Absrel**) [8], and the area under the receiver operating characteristic (**AUROC**) as our evaluation metrics for the confidence estimator. We also follow [47] to regard the depth prediction of *each single pixel* to be correct ($C = 1$) if the relative difference between the depth prediction and the ground truth is less than 25%. Correspondingly, we also regard the depth prediction of *an input image* to be correct ($C = 1$) if the average relative difference among all its pixels (w.r.t the ground truth image) is less than 25%.

At the start of every training epoch, we randomly select 60% of data from D^C to construct the first subset D_1^C, and use the remaining as the second subset D_2^C. On the other hand, D^I is clustered into 6 clusters (i.e., $N = 6$), and one cluster is randomly selected to be D_1^I. We ablate these decisions in Sec. 4.3. During training, we set the learning rate (α) for *virtual training* to $5e-4$, and the learning rate (β) for meta-optimization to $1e-4$.

Experiment Results. In both experiments as shown in Tab. 1 and Tab. 2, we compare our framework with both existing confidence estimation methods and other representative methods on tackling label imbalance problem (Reweight and Resample [3]) and improving out-of-distribution data generalization (Dropout [42], Focal loss [29], and Mixup [48]). Besides, we also compare with the combination of Resample [3] and Mixup [48], which have been shown to effectively tackle label imbalance problem and improving out-of-distribution data generalization respectively. We reimplement all these methods on the same backbone (densenet161) as SLURP and ours. Furthermore, to better assess the effectiveness of our framework in tackling either of the two problems individually, we also

assess the following two variants of our framework, i.e., the variant (**tackling label imbalance only**) that only constructs virtual training and testing sets w.r.t. distribution of the correctness label C, and the variant (**tackling out-of-distribution inputs only**) that only constructs virtual training and testing sets w.r.t. distribution of the data input I.

As shown in Tab. 1, as compared to all other methods, our confidence estimator that is trained on KITTI Eigen-split training set achieves the *best performance across all metrics and testing scenarios*, including both the KITTI Eigen-split testing set from the same dataset and Cityscapes testing set from a different dataset. This demonstrates that, by training the confidence estimator towards improvements on both qualities with our framework, the confidence estimator can become more reliable. Besides, as shown in Tab. 1, both variants of our framework improve the performance of the SLURP baseline, demonstrating that both individual set construction methods can lead to a more reliable confidence estimator, through our *virtual training and testing* scheme.

In Tab. 2, we evaluate the reliability of our confidence estimator under different weather conditions, and report results on the testing sets of Cityscapes, Foggy Cityscapes-DBF and Rainy Cityscapes, where three different weather severity levels are reported for the latter two datasets. We highlight that, under different weather conditions with different severity levels, there are fluctuating degrees of distribution shifts (of C and I) as compared to the Cityscapes training set that are challenging for a confidence estimator to handle. Our framework *outperforms all other methods on all reported metrics*, demonstrating that our framework effectively leads the confidence estimator to learn knowledge that can generalize to diverse distributions.

4.2 Confidence estimation on Image Classification

Settings and Implementation Details. To evaluate our proposed framework on image classification, we follow previous works [19, 5] and conduct experiments on both MNIST [24] and CIFAR-10 [21]. On MNIST, the confidence estimator is trained on MNIST training set and tested on MNIST testing set; and on CIFAR-10, the confidence estimator is trained on CIFAR-10 training set and tested on CIFAR-10 testing set. Note that, in these experiments, even though training/testing are done on same dataset, there are still data distribution shift issues, as shown in [37, 33].

On both datasets, we use the same backbone (Confidnet) as TCP [5], which is described in more detail in the supplementary material. Following [14, 5], we report scores on 4 evaluation metrics: the False positive rate at 95% True positive rate (**FPR-95%-TPR**), the area under the precision-recall curve with respect to $C = 0$ labels (**AUPR-Error**), the area under the precision-recall curve with respect to $C = 1$ labels (**AUPR-Success**), and the area under the receiver operating characteristic (**AUROC**).

Our set construction hyperparameters are set similarly to the monocular depth estimation experiments. At the start of every training epoch, we randomly select 60% of data from D^C to construct D_1^C and designate the rest as D_2^C, while

D^I is split into 6 clusters (i.e., $N = 6$). On MNIST, we set the learning rate (α) for *virtual training* to $1e-4$, and the learning rate (β) for meta-optimization to $1e-4$. On CIFAR-10, we set the learning rate (α) for *virtual training* to $1e-5$, and the learning rate (β) for meta-optimization to $1e-5$.

Results and Analysis. In Tab. 3 we report results using our framework, as well as existing image classification confidence estimation methods and various other representative methods introduced previously (i.e., Reweight and Resample [3], Dropout [42], Focal loss [29], Mixup [48], and Resample [3] + Mixup [48]). As shown in Tab. 3, on both the MNIST testing set and the CIFAR-10 testing set, our framework achieves the *best performance across all metrics*, which demonstrates that our framework can improve confidence estimator performance effectively. The variants of our framework also achieves improvements, which shows the superiority of our framework both in tackling label imbalance problem and improving generalization to distribution shifts in input data. In particular, the variant tackling the label imbalance problem achieves an obvious improvement gain on the AUPR-Error metric which focuses on the performance where $C = 0$.

Table 3. Results on MNIST and CIFAR-10 using same backbone as [5]. When using our variant that tackles label imbalance only, we obtain obvious improvements in the AUPR-Error metric, showing efficacy on tackling cases where $C = 0$.

Dataset	Method	FPR-95%-TPR↓	AUSE-Error↑	AUSE-Success↑	AUROC↑
MNIST [24]	Maximum Class Probability [14]	5.56	35.05	**99.99**	98.63
	MCDropout [11]	5.26	38.50	**99.99**	98.65
	Trust Score [19]	10.00	35.88	99.98	98.20
	Steep Slope Loss [32]	**2.22**	40.86	**99.99**	98.83
	True Class Probability (TCP) [5]	3.33	45.89	**99.99**	98.82
	Balanceed TCP [26]	4.44	43.03	**99.99**	98.67
	TCP + Reweight	7.78	31.67	99.98	98.06
	TCP + Resample [3]	6.67	33.57	99.98	98.32
	TCP + Dropout [42]	3.33	43.05	**99.99**	98.79
	TCP + Focal loss [29]	4.44	42.65	**99.99**	98.73
	TCP + Mixup [48]	4.44	45.73	**99.99**	98.80
	TCP + Resample + Mixup	4.44	45.45	**99.99**	98.78
	TCP + Ours(*tackling label imbalance only*)	**2.22**	46.71	**99.99**	98.87
	TCP + Ours(*tackling out-of-distribution inputs only*)	3.33	45.91	**99.99**	98.85
	TCP + Ours(full)	**2.22**	**47.05**	**99.99**	**98.91**
CIFAR-10 [21]	Maximum Class Probability [14]	47.50	45.36	99.19	91.53
	MCDropout [11]	49.02	46.40	**99.27**	92.08
	Trust Score [19]	55.70	38.10	98.76	88.47
	Steep Slope Loss [32]	**44.69**	50.28	99.26	92.22
	True Class Probability (TCP)[5]	44.94	49.94	99.24	92.12
	Balanceed TCP [26]	45.33	49.79	99.25	92.19
	TCP + Reweight	45.20	49.77	99.25	92.18
	TCP + Resample [3]	45.71	49.81	99.25	92.20
	TCP + Dropout [42]	45.45	49.63	99.25	92.19
	TCP + Focal loss [29]	45.07	49.46	99.24	92.09
	TCP + Mixup [48]	45.33	49.68	99.25	92.18
	TCP + Resample + Mixup	45.20	49.66	99.25	92.18
	TCP + Ours(*tackling label imbalance only*)	44.81	50.27	99.26	92.23
	TCP + Ours(*tackling out-of-distribution inputs only*)	44.81	50.26	99.26	92.23
	TCP + Ours(full)	**44.69**	**50.30**	**99.27**	**92.26**

Experiments on Imagenet. For confidence estimation on image classification, besides evaluating our framework on small scale datasets including MNIST [24] and CIFAR-10 [21] following many previous works [19, 5], we also evaluate our framework on the large-scale dataset Imagenet [7] following [32]. Here, we use the same backbone as [32]. As shown in Tab. 4, after incorporating our framework, we observe a significant performance improvement, which further shows the effectiveness of our method in a large-scale scenario with more classes and larger images, which is more realistic.

4.3 Additional Ablation Studies

In this section and in the supplementary material, we conduct more extensive ablation studies on the monocular depth estimation task, with a confidence esti-

Table 4. Experiment results on Imagenet [7], where our framework is applied on Steep Slope Loss [32], which is the current state-of-the-art. We obtain a significant performance improvement.

Method	FPR-95%-TPR↓	AUSE-Error↑	AUSE-Success↑	AUROC↑
Steep Slope Loss [32]	80.48	10.26	93.01	73.68
Steep Slope Loss + Ours(full)	**76.70**	**10.33**	**94.11**	**78.60**

Table 5. Ablation studies conducted on the effectiveness of the virtual training and testing scheme.

Method	CityScapes [6]			CityScapes Foggy s = 3 [38]			CityScapes Rainy s = 3 [15]		
	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
Joint-training scheme	2.47	5.11	0.867	2.63	4.01	0.829	2.12	3.98	0.869
Meta-learning scheme	0.60	0.62	0.933	0.93	0.58	0.938	1.08	0.80	0.909

mator that is fine-tuned on CityScapes training set. Specifically, our framework is evaluated on the CityScapes testing set, as well as both the Foggy Cityscapes-DBF testing set and Rainy Cityscapes testing set with the highest severity level (i.e., s = 3).

Impact of second-order gradient. In our framework, we update the confidence estimator utilizing the *virtual training and testing* scheme through a second-order gradient $\nabla_\phi\Big(L_{v\_tr}(\phi) + L_{v\_te}\big(\phi - \alpha\nabla_\phi L_{v\_tr}(\phi)\big)\Big)$. To investigate the impact of such a second-order gradient, we compare our framework (**meta-learning scheme**) with a variant (**joint-training scheme**) that still constructs virtual training and testing sets in the same way, but optimizes the confidence estimator through $\nabla_\phi\big(L_{v\_tr}(\phi) + L_{v\_te}(\phi)\big)$ without utilizing the *virtual training and testing* scheme. As shown in Tab. 5, our framework consistently outperforms this variant, which shows effectiveness of the *virtual training and testing* scheme.

5 Conclusion

In this paper, we propose a unified framework that improves the reliability of confidence estimators, through simultaneously improving their performance under label imbalance and their handling of various out-of-distribution data inputs. Through carefully constructing virtual training and testing sets with different distributions w.r.t. both the correctness label C and the data input I, our framework trains the confidence estimator with a *virtual training and testing* scheme and leads it to learn knowledge that is more generalizable to different distributions (w.r.t. both the C and I). To validate the general effectiveness of our framework, we apply our framework to confidence estimation methods on both monocular depth estimation and image classification tasks, and show consistent improvements on both.

Acknowledgement

This work is supported by National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-100E-2020-065), Ministry of Education Tier 1 Grant and SUTD Startup Research Grant.

References

1. Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., Mané, D.: Concrete problems in ai safety. arXiv preprint arXiv:1606.06565 (2016) 1
2. Bai, Y., Jiao, J., Ce, W., Liu, J., Lou, Y., Feng, X., Duan, L.Y.: Person30k: A dual-meta generalization network for person re-identification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2123–2132 (2021) 3, 4
3. 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) 10, 11, 13
4. Chen, J., Liu, F., Avci, B., Wu, X., Liang, Y., Jha, S.: Detecting errors and estimating accuracy on unlabeled data with self-training ensembles. Advances in Neural Information Processing Systems **34** (2021) 3
5. 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) 2, 3, 4, 9, 10, 11, 12, 13
6. 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) 10, 11, 14
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009) 13, 14
8. 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) 10, 11
9. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning. pp. 1126–1135. PMLR (2017) 3, 4
10. Floridi, L.: Establishing the rules for building trustworthy ai. Nature Machine Intelligence **1**(6), 261–262 (2019) 1
11. 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) 3, 4, 10, 11, 13
12. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. International Journal of Robotics Research (IJRR) (2013) 10
13. 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) 3, 4
14. Hendrycks, D., Gimpel, K.: A baseline for detecting misclassified and out-of-distribution examples in neural networks. Proceedings of International Conference on Learning Representations (2017) 3, 4, 12, 13
15. 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, 11, 14
16. Huang, C., Cao, Z., Wang, Y., Wang, J., Long, M.: Metasets: Meta-learning on point sets for generalizable representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8863–8872 (2021) 3, 4

17. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: Proceedings of the IEEE international conference on computer vision. pp. 1501–1510 (2017) 8
18. Janai, J., Güney, F., Behl, A., Geiger, A., et al.: Computer vision for autonomous vehicles: Problems, datasets and state of the art. Foundations and Trends® in Computer Graphics and Vision **12**(1–3), 1–308 (2020) 1
19. Jiang, H., Kim, B., Guan, M., Gupta, M.: To trust or not to trust a classifier. Advances in neural information processing systems **31** (2018) 2, 3, 12, 13
20. Kendall, A., Gal, Y.: What uncertainties do we need in bayesian deep learning for computer vision? Advances in neural information processing systems **30** (2017) 10, 11
21. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009) 2, 12, 13
22. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25**, 1097–1105 (2012) 1
23. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. Advances in neural information processing systems **30** (2017) 10, 11
24. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998) 2, 12, 13
25. Lee, J.H., Han, M.K., Ko, D.W., Suh, I.H.: From big to small: Multi-scale local planar guidance for monocular depth estimation. arXiv preprint arXiv:1907.10326 (2019) 1
26. Li, B., Zheng, Z., Zhang, C.: Identifying incorrect classifications with balanced uncertainty. arXiv preprint arXiv:2110.08030 (2021) 2, 3, 4, 13
27. Li, B., Wu, F., Lim, S.N., Belongie, S., Weinberger, K.Q.: On feature normalization and data augmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12383–12392 (2021) 8
28. 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) 3, 4
29. 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) 4, 10, 11, 13
30. Linda, O., Vollmer, T., Manic, M.: Neural network based intrusion detection system for critical infrastructures. In: 2009 international joint conference on neural networks. pp. 1827–1834. IEEE (2009) 1
31. Liu, S., Deng, W.: Very deep convolutional neural network based image classification using small training sample size. In: 2015 3rd IAPR Asian conference on pattern recognition (ACPR). pp. 730–734. IEEE (2015) 2
32. Luo, Y., Wong, Y., Kankanhalli, M.S., Zhao, Q.: Learning to predict trustworthiness with steep slope loss. Advances in Neural Information Processing Systems **34** (2021) 2, 3, 4, 13, 14
33. 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) 2, 8, 9, 12
34. Moon, J., Kim, J., Shin, Y., Hwang, S.: Confidence-aware learning for deep neural networks. In: international conference on machine learning. pp. 7034–7044. PMLR (2020) 3

35. 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) 2, 3, 4
36. Qiu, X., Miikkulainen, R.: Detecting misclassification errors in neural networks with a gaussian process model. arXiv preprint arXiv:2010.02065 (2020) 3
37. 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) 12
38. 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) 10, 11, 14
39. Sanz, J.A., Galar, M., Jurio, A., Brugos, A., Pagola, M., Bustince, H.: Medical diagnosis of cardiovascular diseases using an interval-valued fuzzy rule-based classification system. Applied Soft Computing **20**, 103–111 (2014) 1
40. Shafaei, S., Kugele, S., Osman, M.H., Knoll, A.: Uncertainty in machine learning: A safety perspective on autonomous driving. In: International Conference on Computer Safety, Reliability, and Security. pp. 458–464. Springer (2018) 1
41. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1409.1556 2
42. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research **15**(1), 1929–1958 (2014) 10, 11, 13
43. Tomani, C., Gruber, S., Erdem, M.E., Cremers, D., Buettner, F.: Post-hoc uncertainty calibration for domain drift scenarios. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10124–10132 (2021) 2, 3, 4
44. Tsiligkaridis, T.: Failure prediction by confidence estimation of uncertainty-aware dirichlet networks. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 3525–3529. IEEE (2021) 3
45. 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) 10
46. 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) 3, 4
47. 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) 2, 3, 4, 9, 10, 11
48. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: International Conference on Learning Representations (2018) 10, 11, 13