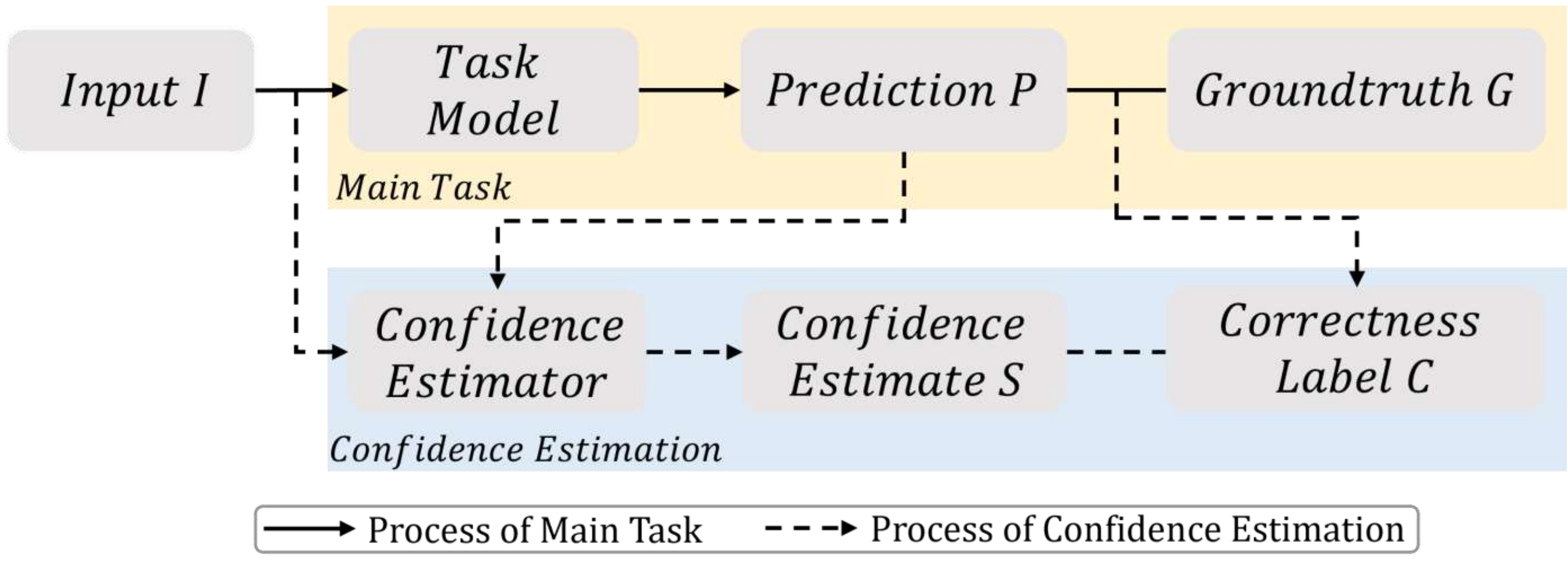
Improving the Reliability for Confidence Estimation

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

[1]Singapore University of Technology and Design & [2]Adobe Research

* Indicates equal contribution, ** Indicates corresponding author

What is Confidence Estimation?



Confidence estimation aims to evaluate the confidence of the model's prediction during deployment.

Important Qualities

- A reliable confidence estimator should *perform well under label imbalance*.
- A reliable confidence estimator should *hold the ability to handle out-of-distribution data inputs*.

Aim and Contribution

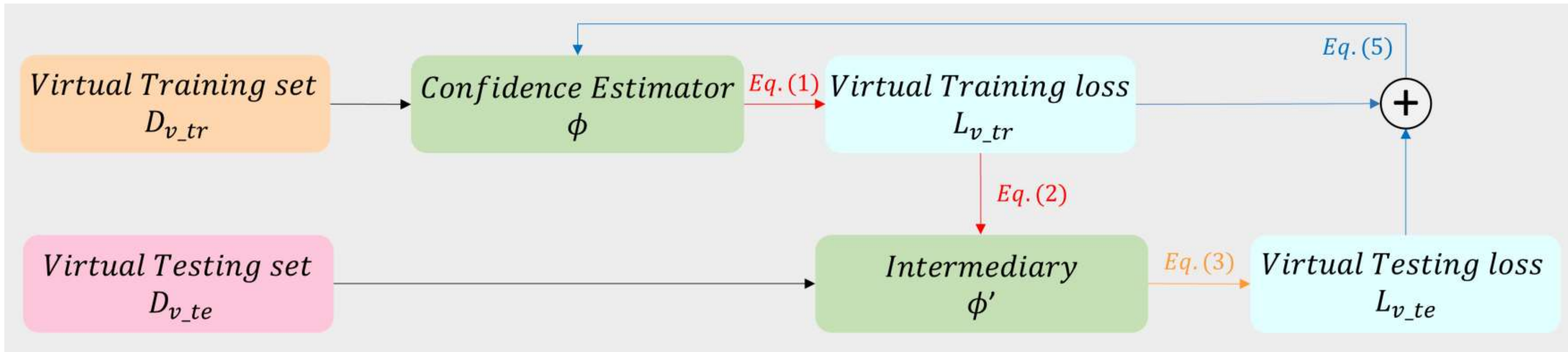
Aim:

- In this paper, we aim to improve the reliability of our confidence estimator in terms of both the above-mentioned qualities.
- Common point of the above qualities: They are acquired when the confidence estimator learns to generalize to diverse distributions.

Contributions:

- We propose a novel framework, which incorporates meta-learning to learn a confidence estimator to produce confidence estimates more reliably.
- 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.

Virtual Training and Testing



- **Virtual Training**: 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).
- **Virtual Testing**: 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).
- **Meta Optimization (Actual update)**: 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.

Set Construction

Constructing sets for correctness label C:

- Step (C1). At the start of each epoch, we first randomly split D^c into two subsets D_1^c and D_2^c.
- 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}.

Constructing sets for data input I:

- 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 .
- Step (I2). Then at the start of every even-numbered iteration, we first randomly select a batch of data from the selected cluster D_1^I 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}.

Overall Training Scheme

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

Experiments

We conduct experiments across various computer vision tasks, including:

- image classification
- monocular depth estimation

Below, we show part of our experiment results and visualizations.

Method	KITTI [12, 45, 8]			CityScapes [6]		
	AUSE-RMSE↓	AUSE-Absrel↓	AUROC↑	AUSE-RMSE↓	AUSE-Absrel↓	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**

Experiment results of confidence estimation on monocular depth estimation on KITTI dataset.

Visualizations w.r.t the data with incorrect task model predictions from CIFAR-10 dataset.