Rethinking Importance Weighting for Deep Learning under Distribution Shift

Tongtong Fang*[1] Nan Lu*[1,2] Gang Niu[2] Masashi Sugiyama[2,1]




* Equal Contribution

[1] Univ. of Tokyo, Japan

[2] RIKEN, Japan

NeurIPS 2020 Spotlight Presentation

About me

- Nan LU
- Ph.D. student at the University of Tokyo
- Research interests
  - Weakly supervised learning
    - Positive-unlabeled classification
    - Unlabeled-unlabeled classification
    - Learning under distribution shift
  - Deep learning
  - Privacy-preserving learning

Motivations

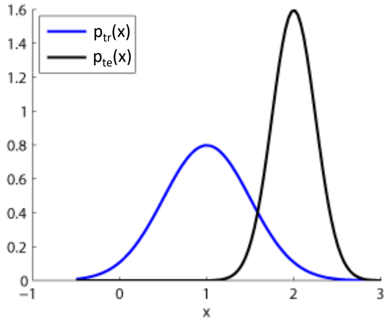
Distribution Shift Almost Everywhere

- Distribution shift: the training data distribution differs from the test one $p_{tr}(\boldsymbol{x}, y) \neq p_{te}(\boldsymbol{x}, y)$

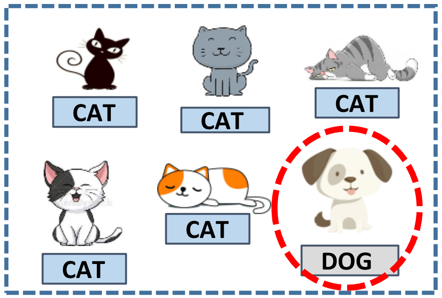
Distribution Shift Almost Everywhere

- Distribution shift: the training data distribution differs from the test one $p_{tr}(\boldsymbol{x}, y) \neq p_{te}(\boldsymbol{x}, y)$

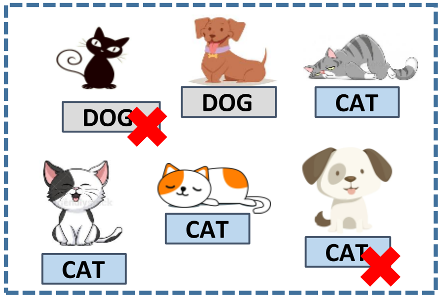
- Covariate shift $p_{tr}(\boldsymbol{x}) \neq p_{te}(\boldsymbol{x})$



- Class-prior shift $p_{tr}(y) \neq p_{te}(y)$



- Label noise $p_{tr}(y|\boldsymbol{x}) \neq p_{te}(y|\boldsymbol{x})$

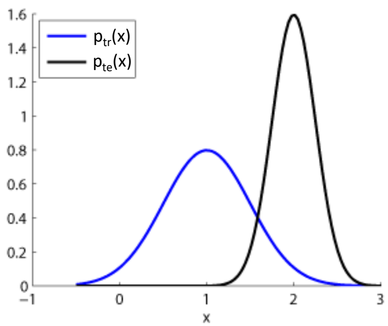


Distribution Shift Almost Everywhere

- Distribution shift: the training data distribution differs from the test one $p_{tr}(\boldsymbol{x}, y) \neq p_{te}(\boldsymbol{x}, y)$

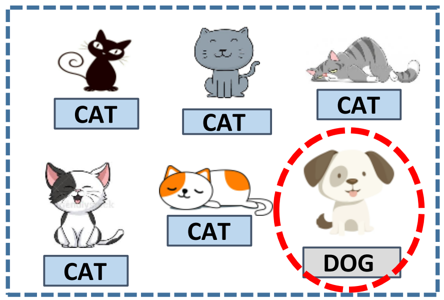
- Covariate shift

$$p_{tr}(\boldsymbol{x}) \neq p_{te}(\boldsymbol{x})$$



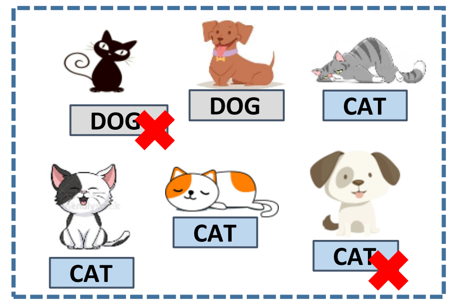
- Class-prior shift

$$p_{tr}(y) \neq p_{te}(y)$$



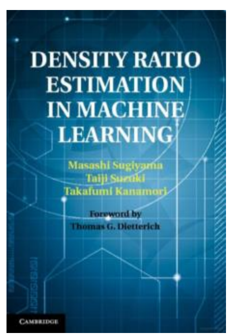
- Label noise

$$p_{tr}(y|\boldsymbol{x}) \neq p_{te}(y|\boldsymbol{x})$$

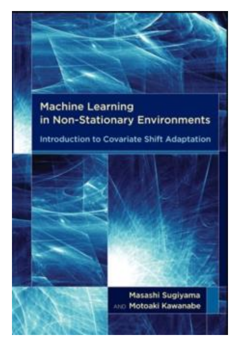


More than 200 top conference papers in the last two decades!

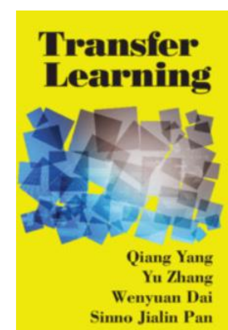
The MIT Press, 2009



Cambridge University Press, 2012



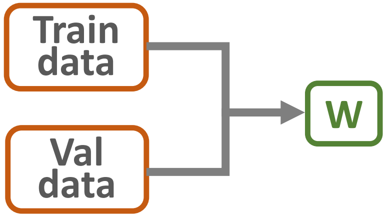
The MIT press, 2012



Cambridge University Press, 2020

Powerful Tool: Importance Weighting (IW)

- Step one: weight estimation (WE)†

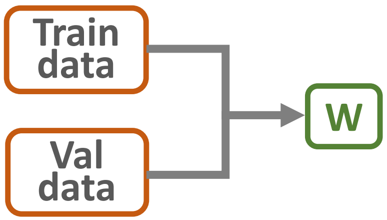


† aka density ratio estimation

$$w^*(\boldsymbol{x}, y) = p_{te}(\boldsymbol{x}, y)/p_{tr}(\boldsymbol{x}, y)$$

Powerful Tool: Importance Weighting (IW)

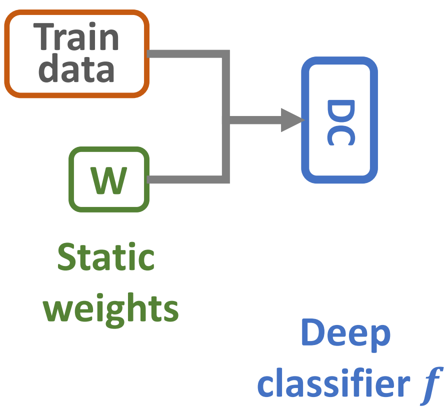
- Step one: weight estimation (WE)[†]



† aka density ratio estimation

$$w^*(\boldsymbol{x}, y) = p_{te}(\boldsymbol{x}, y)/p_{tr}(\boldsymbol{x}, y)$$

- Step two: weighted classification (WC)



$$\mathbb{E}_{p_{te}(\boldsymbol{x},y)}[\boldsymbol{f}(\boldsymbol{x}, y)] = \mathbb{E}_{p_{tr}(\boldsymbol{x},y)}[w^*(\boldsymbol{x}, y)\boldsymbol{f}(\boldsymbol{x}, y)]$$

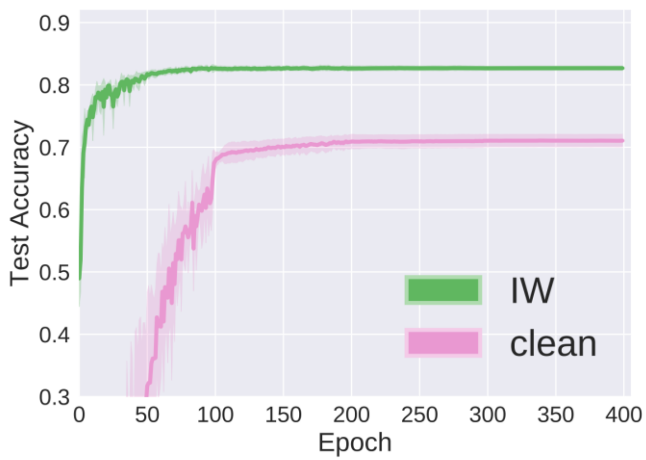
Goal of Our Work

- IW is the common practice of non-deep learning under distribution shift [1,2,3]

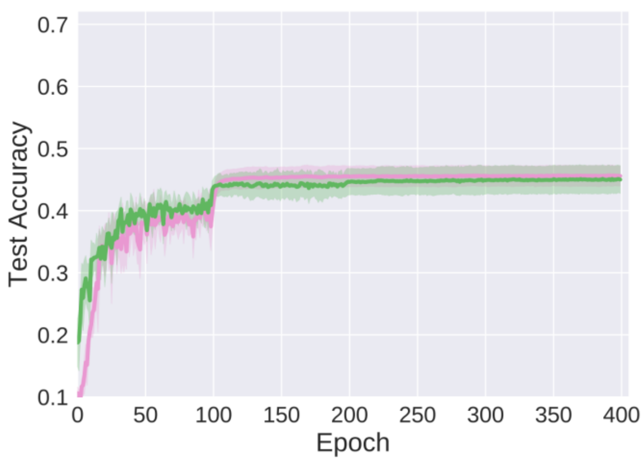
[1] Density ratio estimation in machine learning. Cambridge University Press, 2012.
[2] Dataset shift in machine learning. The MIT Press, 2009.
[3] Machine learning in non-stationary environments: Introduction to covariate shift adaptation. The MIT press, 2012.

Goal of Our Work

- IW is the common practice of non-deep learning under distribution shift [1,2,3]
- But IW cannot work well on complex data



IW is still OK on Fashion-MNIST



IW fails on CIFAR-10

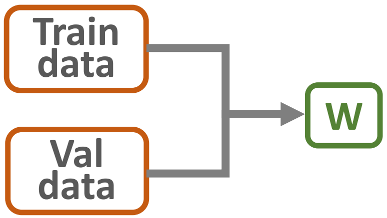
- Clean: use 1,000 training data, with no distribution shift
- IW: use all training data (60,000/50,000) under 0.3 pair-flip label noise

[1] Density ratio estimation in machine learning. Cambridge University Press, 2012.
[2] Dataset shift in machine learning. The MIT Press, 2009.
[3] Machine learning in non-stationary environments: Introduction to covariate shift adaptation. The MIT press, 2012.

Methods

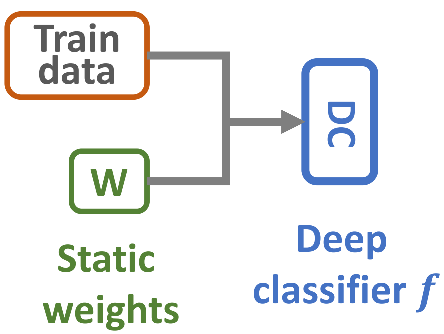
Rethinking Importance Weighting (IW)

- Step one: weight estimation (WE)



$$w^*(\boldsymbol{x}, y) = p_{te}(\boldsymbol{x}, y)/p_{tr}(\boldsymbol{x}, y)$$

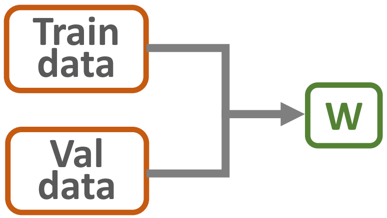
- Step two: weighted classification (WC)



$$\mathbb{E}_{p_{te}(\boldsymbol{x},y)}[\boldsymbol{f}(\boldsymbol{x}, y)] = \mathbb{E}_{p_{tr}(\boldsymbol{x},y)}[w^*(\boldsymbol{x}, y)\boldsymbol{f}(\boldsymbol{x}, y)]$$

Rethinking Importance Weighting (IW)

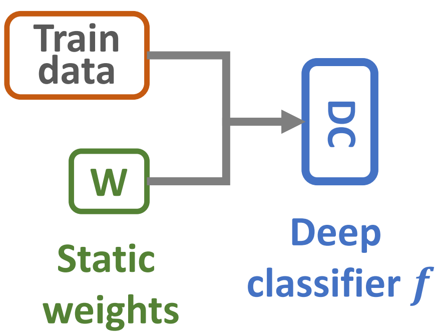
- Step one: weight estimation (WE)



$$w^*(\boldsymbol{x}, y) = p_{te}(\boldsymbol{x}, y)/p_{tr}(\boldsymbol{x}, y)$$

Difficult to boost the expressive power of WE

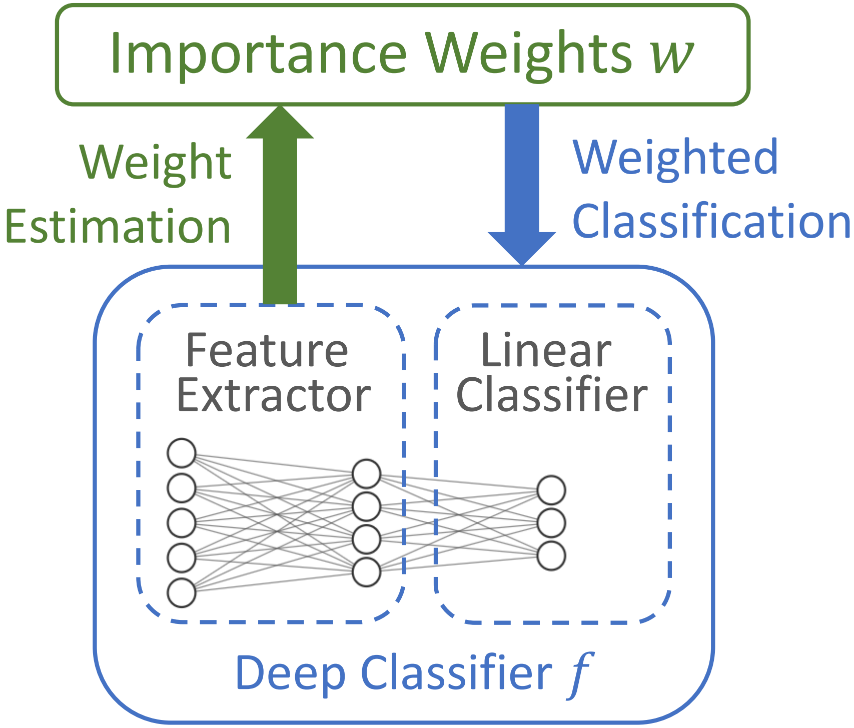
- Step two: weighted classification (WC)



$$\mathbb{E}_{p_{te}(\boldsymbol{x},y)}[\boldsymbol{f}(\boldsymbol{x}, y)] = \mathbb{E}_{p_{tr}(\boldsymbol{x},y)}[w^*(\boldsymbol{x}, y)\boldsymbol{f}(\boldsymbol{x}, y)]$$

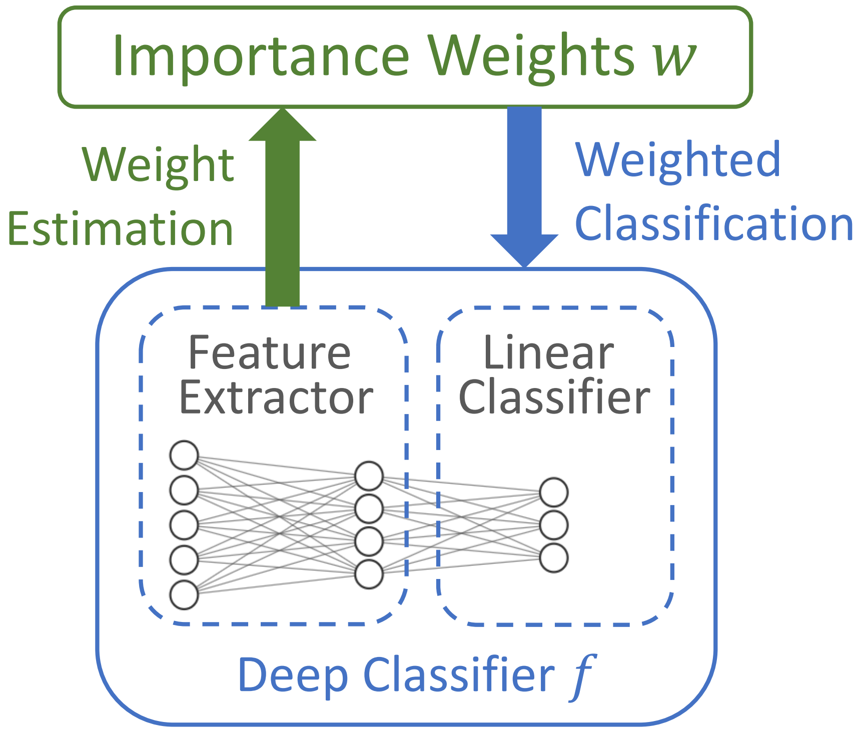
Most powerful deep models are hard to train with the WE optimizations

Circular Dependency



- Idea: boost the expressive power of WE by a feature extractor created from f

Circular Dependency



- Idea: Boost by an external feature extractor inside f
- Causality dilemma:
  - Need w to train f
  - Need a trained f to estimate w
  - Chicken or egg?

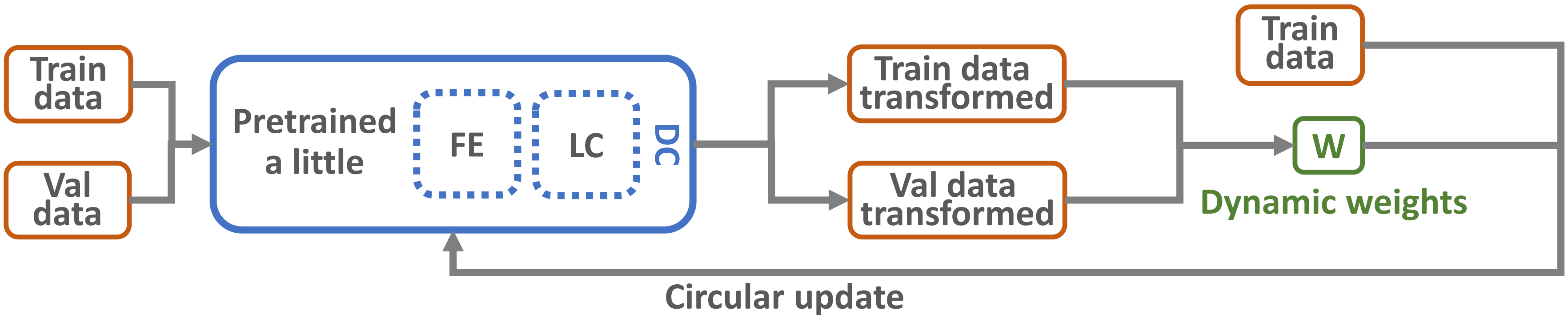
Non-linear Transformation of Data

Theorem 1. For a fixed, deterministic, and invertible transformation $\pi : (\boldsymbol{x}, y) \mapsto \boldsymbol{z}$, let $p_{\mathrm{tr}}(\boldsymbol{z})$ and $p_{\mathrm{te}}(\boldsymbol{z})$ be the probability density functions (PDFs) induced by $p_{\mathrm{tr}}(\boldsymbol{x}, y)$, $p_{\mathrm{te}}(\boldsymbol{x}, y)$ and π. Then,

$$w^*(\boldsymbol{x}, y) = \frac{p_{\mathrm{te}}(\boldsymbol{x}, y)}{p_{\mathrm{tr}}(\boldsymbol{x}, y)} = \frac{p_{\mathrm{te}}(\boldsymbol{z})}{p_{\mathrm{tr}}(\boldsymbol{z})} = w^*(\boldsymbol{z}).$$

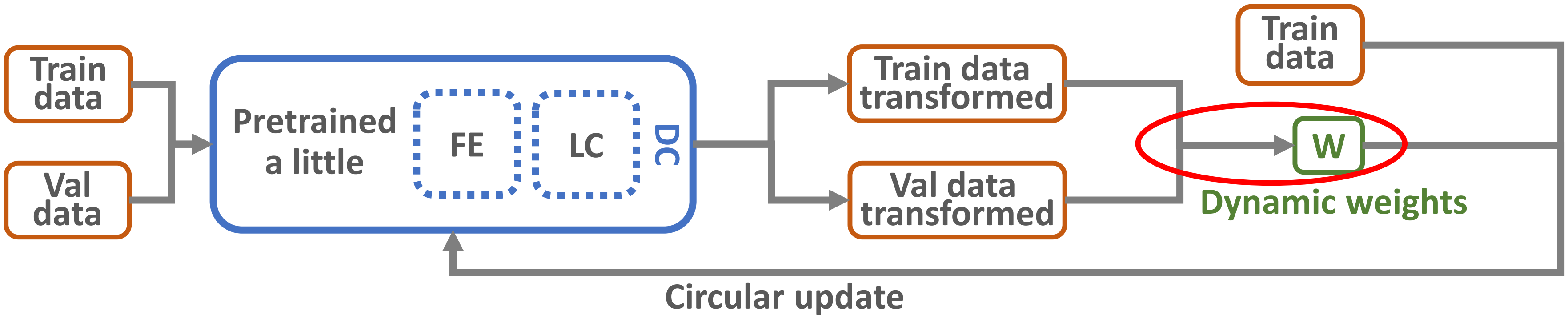
If π is from part of $\boldsymbol{f}$, $\boldsymbol{f}$ must be a reasonably good classifier so that π compresses data back to a manifold.

Dynamic Importance Weighting (DIW)



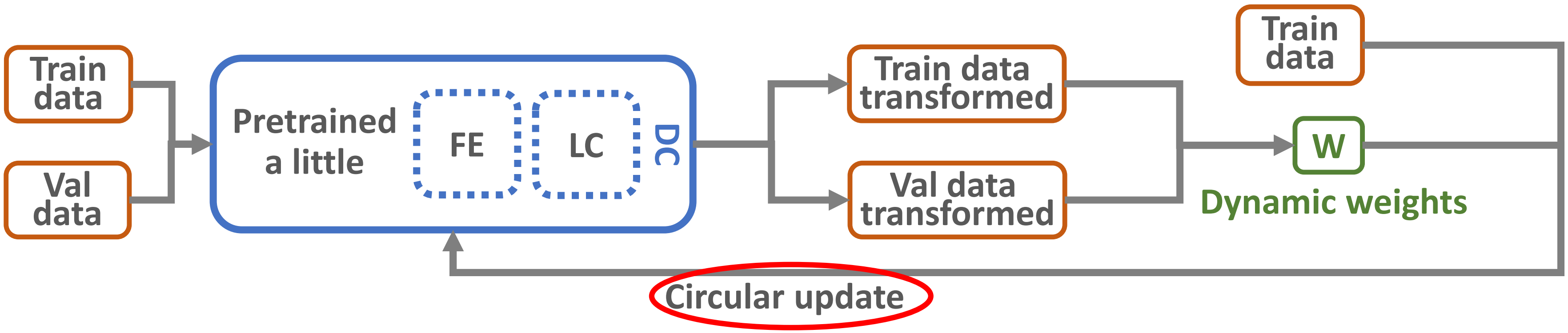
- End-to-end solution
- Train a deep classifier (DC) from **weighted** training data and create a feature extractor (FE) from DC
- Meanwhile perform weight estimation on the data transformed by FE in a seamless manner

Dynamic Importance Weighting (DIW)



- End-to-end solution
- Train a deep classifier (DC) from **weighted** training data and create a feature extractor (FE) from DC
- Meanwhile perform weight estimation on the data transformed by FE in a seamless manner

Dynamic Importance Weighting (DIW)



- End-to-end solution
- Train a deep classifier (DC) from **weighted** training data and create a feature extractor (FE) from DC
- Meanwhile perform weight estimation on the data transformed by FE in a seamless manner

Dynamic Importance Weighting (DIW)

Algorithm 1 Dynamic importance weighting (in a mini-batch).

Require: a training mini-batch $\mathcal{S}^{\mathrm{tr}}$, a validation mini-batch $\mathcal{S}^{\mathrm{v}}$, the current model $\boldsymbol{f}_{\theta_t}$

Hidden-layer-output transformation version:

`forward` the input parts of $\mathcal{S}^{\mathrm{tr}}$ & $\mathcal{S}^{\mathrm{v}}$
`retrieve` the hidden-layer outputs $\mathcal{Z}^{\mathrm{tr}}$ & $\mathcal{Z}^{\mathrm{v}}$
`partition` $\mathcal{Z}^{\mathrm{tr}}$ & $\mathcal{Z}^{\mathrm{v}}$ into $\{\mathcal{Z}_y^{\mathrm{tr}}\}_{y=1}^k$ & $\{\mathcal{Z}_y^{\mathrm{v}}\}_{y=1}^k$
for $y = 1, \ldots, k$ **do**
    `match` $\mathcal{Z}_y^{\mathrm{tr}}$ & $\mathcal{Z}_y^{\mathrm{v}}$ to obtain $\mathcal{W}_y$
    `multiply` all $w_i \in \mathcal{W}_y$ by w_y^*
end for
`compute` the loss values of $\mathcal{S}^{\mathrm{tr}}$ as $\mathcal{L}^{\mathrm{tr}}$
`weight` the empirical risk $\widehat{R}(\boldsymbol{f}_\theta)$ by $\{\mathcal{W}_y\}_{y=1}^k$
`backward` $\widehat{R}(\boldsymbol{f}_\theta)$ and update θ

Loss-value transformation version:

`forward` the input parts of $\mathcal{S}^{\mathrm{tr}}$ & $\mathcal{S}^{\mathrm{v}}$
`compute` the loss values as $\mathcal{L}^{\mathrm{tr}}$ & $\mathcal{L}^{\mathrm{v}}$
`match` $\mathcal{L}^{\mathrm{tr}}$ & $\mathcal{L}^{\mathrm{v}}$ to obtain $\mathcal{W}$
`weight` the empirical risk $\widehat{R}(\boldsymbol{f}_\theta)$ by $\mathcal{W}$
`backward` $\widehat{R}(\boldsymbol{f}_\theta)$ and update θ

Practical Choices of Data Transformation

Hidden-layer-output transformation

- Estimate $w_y^* = p_{\mathrm{te}}(y)/p_{\mathrm{tr}}(y)$
- Partition training and val data according to y
- Invoke weight estimation k times on k partitions

$$\frac{p_{\mathrm{te}}(\boldsymbol{x},y)}{p_{\mathrm{tr}}(\boldsymbol{x},y)} = \frac{p_{\mathrm{te}}(y)\cdot p_{\mathrm{te}}(\boldsymbol{x}|y)}{p_{\mathrm{tr}}(y)\cdot p_{\mathrm{tr}}(\boldsymbol{x}|y)} = w_y^* \cdot \frac{p_{\mathrm{te}}(\boldsymbol{x}|y)}{p_{\mathrm{tr}}(\boldsymbol{x}|y)} = w_y^* \cdot \frac{p_{\mathrm{te}}(\boldsymbol{z}|y)}{p_{\mathrm{tr}}(\boldsymbol{z}|y)}$$

Practical Choices of Data Transformation

Hidden-layer-output transformation

- Estimate $w_y^* = p_{\mathrm{te}}(y)/p_{\mathrm{tr}}(y)$
- Partition training and val data according to y
- Invoke weight estimation k times on k partitions

$$\frac{p_{\mathrm{te}}(\boldsymbol{x},y)}{p_{\mathrm{tr}}(\boldsymbol{x},y)} = \frac{p_{\mathrm{te}}(y)\cdot p_{\mathrm{te}}(\boldsymbol{x}|y)}{p_{\mathrm{tr}}(y)\cdot p_{\mathrm{tr}}(\boldsymbol{x}|y)} = w_y^* \cdot \frac{p_{\mathrm{te}}(\boldsymbol{x}|y)}{p_{\mathrm{tr}}(\boldsymbol{x}|y)} = w_y^* \cdot \frac{p_{\mathrm{te}}(\boldsymbol{z}|y)}{p_{\mathrm{tr}}(\boldsymbol{z}|y)}$$

Loss-value transformation

- Find a set of weights $\mathcal{W} = \{w_i\}_{i=1}^{n_{\mathrm{tr}}}$
  such that for $\ell(\boldsymbol{f}_\theta(\boldsymbol{x}), y)$,

$$\frac{1}{n_{\mathrm{v}}} \sum_{i=1}^{n_{\mathrm{v}}} \ell(\boldsymbol{f}_\theta(\boldsymbol{x}_i^{\mathrm{v}}), y_i^{\mathrm{v}})\big|_{\theta=\theta_t} \approx \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} w_i \ell(\boldsymbol{f}_\theta(\boldsymbol{x}_i^{\mathrm{tr}}), y_i^{\mathrm{tr}})\big|_{\theta=\theta_t}$$

Distribution Matching

- Kernel mean matching (KMM) [1]

We minimize $\boldsymbol{w}^\top \boldsymbol{K} \boldsymbol{w} - 2\boldsymbol{k}^\top \boldsymbol{w} + \mathrm{Const.}$,

subject to $0 \leq w_i \leq B$ and

$|\frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} w_i - 1| \leq \epsilon$

$\boldsymbol{w}$: weight vector

$\boldsymbol{K}_{ij} = k(\boldsymbol{z}_i^{\mathrm{tr}}, \boldsymbol{z}_j^{\mathrm{tr}})$

$\boldsymbol{k}_i = \frac{n_{\mathrm{tr}}}{n_{\mathrm{v}}} \sum_{j=1}^{n_{\mathrm{v}}} k(\boldsymbol{z}_i^{\mathrm{tr}}, \boldsymbol{z}_j^{\mathrm{v}})$

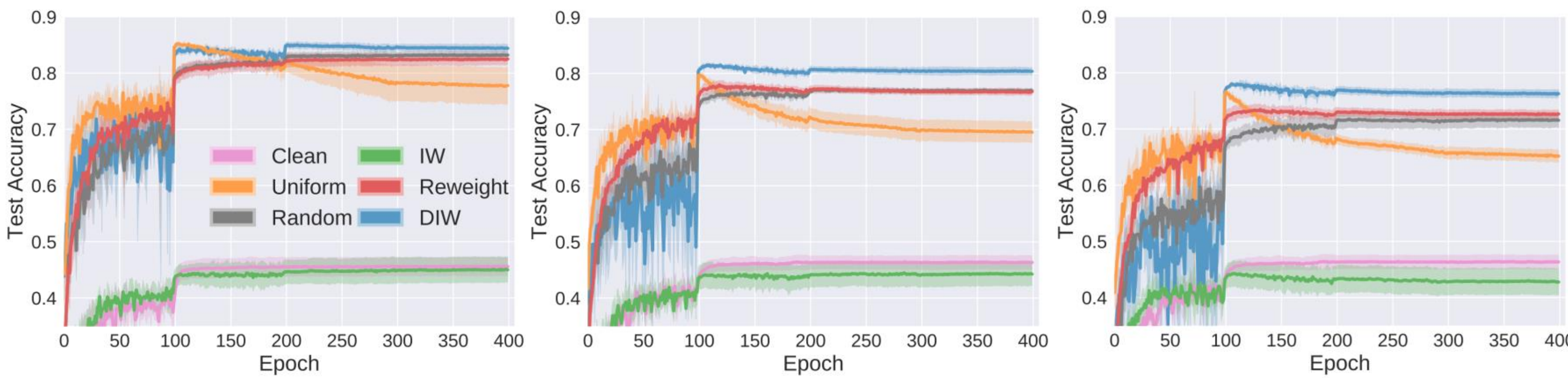
- In IW, KMM is performed on all training data within one class
- In DIW, KMM is performed on transformed data in every mini-batch

[1] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. Smola. Correcting sample selection bias by unlabeled data. In NeurIPS, 2007.

Experiments

Label-noise Experiments

- Setting: $p_{tr}(\boldsymbol{x}) = p_{te}(\boldsymbol{x}), p_{tr}(y|\boldsymbol{x}) \neq p_{te}(y|\boldsymbol{x})$
- Classification results on CIFAR-10



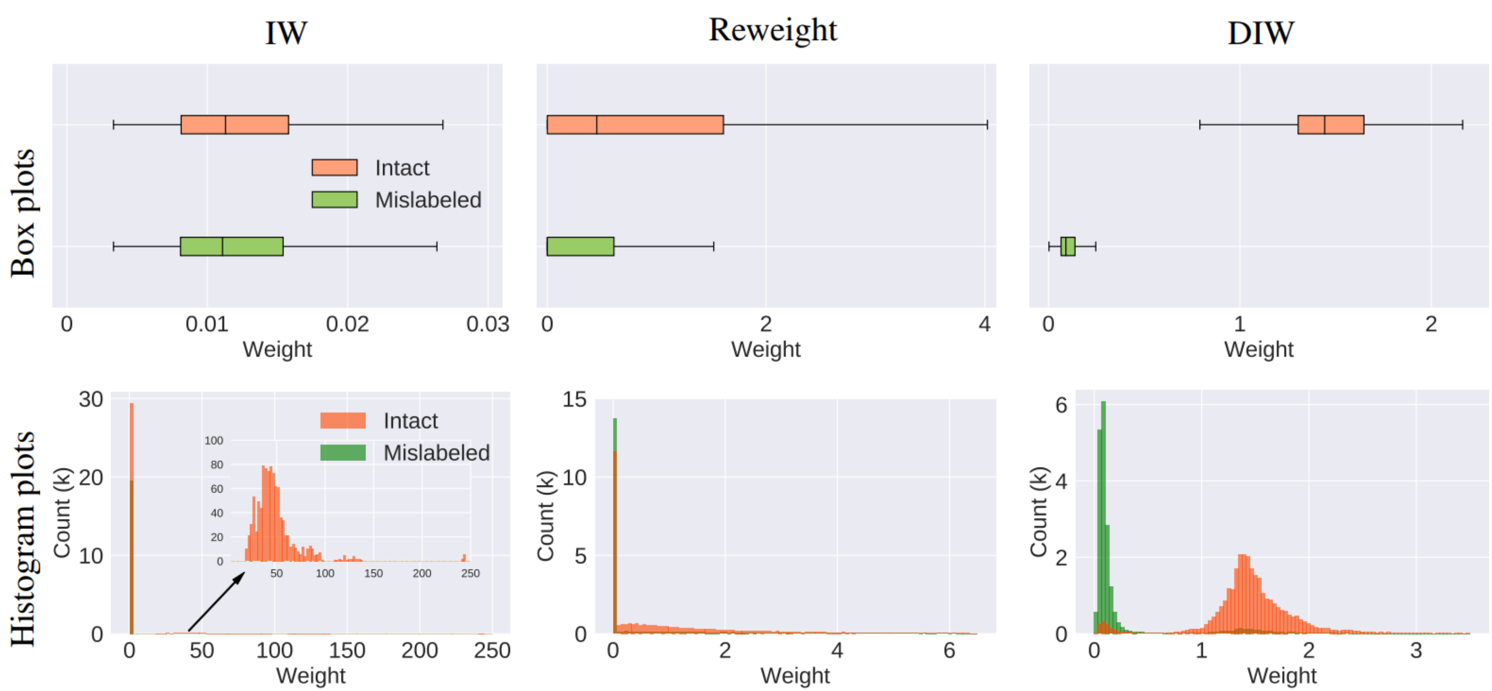
0.3 pair flip | 0.4 symmetric flip | 0.5 symmetric flip

- Base model: ResNet-32
- Optimizer: Adam

- **DIW**: dynamic importance weighting
- **IW**: importance weighting

Label-noise Experiments

- Setting: $p_{tr}(\boldsymbol{x}) = p_{te}(\boldsymbol{x}), p_{tr}(y|\boldsymbol{x}) \neq p_{te}(y|\boldsymbol{x})$
- Statistics of weight distributions on CIFAR-10 under 0.4 symmetric flip



Many More Experiments in the Paper

- Label-noise experiments on Fashion-MNIST & CIFAR-100
- Class-prior-shift experiments on Fashion-MNIST
- Many ablation studies
  - DIW design options: updating/pretraining FE, choices of data transformation
  - Denoising effect analysis

Take-home Messages

- For deep learning under distribution shift, IW suffers from a circular dependency
- To avoid this issue, dynamic IW (DIW) is proposed as an end-to-end solution
  - introduce feature extractors
  - embed IW in every mini-batch
- Many algorithm design options of DIW are available

Thanks for your attention!