Supplementary Material

Detecting Errors and Estimating Accuracy on Unlabeled Data with Self-training Ensembles

In Section A, we have some further discussions to clarify some important points about our work. In Section B, we present our theoretical results, their proofs and the discussions on the benefit of using ensembles. In Section C, we describe the detailed settings for the experiments and also present some additional experimental results.

A Further Discussions

A.1 Comparisons with Proxy Risk

Our work and proxy risk work have the following similarities:

1. Both proxy risk and our framework train check models to estimate the accuracy of the pre-trained model f on the unlabeled test dataset U and also identify misclassified points in U;
2. Both proxy risk and one instance of our framework (Algorithm 3) use domain-invariant representations (DIR) to improve the accuracy of the check models on the target domain.

However, they also have the following differences:

1. The key ideas are different. The idea of Proxy Risk is to find a check model with maximum disagreement in the good hypothesis class (the set of hypotheses that achieve small DIR loss). Our idea is to **increase** the disagreement on the **mis-classified points** in each iteration of self-training, and mis-classified data points are identified by either accurate prediction or diversity using ensemble;
2. The training objectives are different: 1) Proxy Risk's objective is applied on the whole unlabeled test set U. Ours is only on the selected subset of U (i.e., the currently identified mis-classified points); 2) The terms in the objective to encourage disagreement are different. Proxy risk tries to maximize the disagreement between the model f and the check model h directly on the entire unlabeled test set U while maintaining a small DIR loss (corresponding to the term $-\mathbb{E}_{x\in U_X}\ell(f(x),h'(x))$ in their objective). In contrast, our method encourages disagreement via fitting the check models to the pseudo-labeled dataset R (corresponding to the term $\mathbb{E}_{(x,y)\in R}\ell(h(x),y)$ in our objective). The two terms $\mathbb{E}_{(x,y)\in R}\ell(h(x),y)$ and $-\mathbb{E}_{x\in U_X}\ell(f(x),h'(x))$ are different. For multi-class classification, disagreeing with the pre-trained model is not equivalent to agreeing with the pseudo-labels. This is because the check model can predict some labels different from both the pre-trained model's prediction $f(x)$ and the pseudo-label $\tilde{y}_x$. For example, suppose there are 3 classes, and suppose the pre-trained model predicts $f(x)$= class 0, the pseudo-label $\tilde{y}_x$=class 1. Our term is ensuring $h(x)$ to be class 1, while their term is ensuring $h'(x)$ to be either class 1 or class 2. As has been pointed out in our previous reply, our objective is more specific, and this leads to increasing disagreement and also potentially better prediction from the ensemble, which then leads to the success of the self-training;
3. The implementation of the objectives are also different. The proxy risk method uses L_2 norm: $-\mathbb{E}_{x\sim U_X}[\ell(f(x),h'(x))] = \mathbb{E}_{x\sim U_X}[-\|\bar{h}'(x)-\bar{f}(x)\|_2]$ [3] while our method uses cross entropy loss $\mathbb{E}_{(x,y)\in R}[\ell(h(x),y)] = \mathbb{E}_{(x,y)\in R}[-\log \bar{h}(x)_y]$. Here, $\bar{h}'$ is the softmax output of h (similarly for $\bar{f}$ and $\bar{h}$).

A.2 Comparisons with Standard Self-training and Ensemble

Our self-training ensembles method is different from the standard self-training and ensemble. The differences are:

[3] See their code: https://github.com/chingyaoc/estimating-generalization

1. The goals are different. Standard self-training+ensemble aims to get accurate predictions. Ours is to increase disagreement on mis-classified points by either getting accurate predictions or getting diversity.
2. The techniques are also different, due to the different goals. (1) For each identified mis-classified point, we assign a pseudo-label that is different from the prediction of the pre-trained model, but we do not need the pseudo-labels to be correct, as our goal is disagreement. In contrast, standard methods typically hope the pseudo-labels are correct. (2) We only assign pseudo-labels to the currently identified set of mis-classified points, while standard methods typically assign pseudo-labels for all unlabeled points. This is because we want disagreement on mis-classified points rather than on all points, and we would like to assign pseudo-labels to be different from the prediction of the pre-trained model on mis-classified points.

B Complete Proofs and Discussions for Section 5

We first present the proof for our main theorem, and then provide some more discussion on the benefit of using ensembles.

B.1 Provable Guarantees of the Framework

We first prove a technical lemma for constructing R and then use it to prove the theorem.

First recall the key notions.

Suppose on points where f is correct, the ensemble models are also approximately correct. Let ν denote the average probability of $h \sim \mathcal{T}$ making error on the test points where the pre-trained model f is correct:

$$\nu := \Pr_{(x,y_x)\sim U, h\sim\mathcal{T}}[h(x) \neq y_x | f(x) = y_x]. \tag{5}$$

Suppose the ensemble training with regularization on the pseudo-labeled data R will make the ensemble models disagree with f on R_X. Let γ denote the average probability of $h \sim \mathcal{T}$ agreeing with f on the test points in R_X:

$$\gamma := \Pr_{x\sim R_X, h\sim\mathcal{T}}\{h(x) = f(x)\}. \tag{6}$$

Suppose G_X is the points in $W_X \setminus R_X$ on which the ensemble agree with the true label y_x with more than $1-\nu$ probability, i.e.,

$$G_X := \{x \in W_X \setminus R_X : \Pr_{h\sim\mathcal{T}}\{h(x) = y_x\} \geq 1-\nu\}. \tag{7}$$

That is, G_X are the points where the ensemble will have correct prediction with high confidence. We would like the ensemble to have large diversity on the remaining points in $W_X \setminus R_X$. Define the diversity there to be

$$B_X := W_X \setminus (R_X \cup G_X), \tag{8}$$

$$\sigma^2 := \mathbb{E}[\sigma_x^2 | x \in B_X]. \tag{9}$$

Lemma 1. *Define*

$$B_\eta := \min\{\sigma^2, 1-\nu^2\}. \tag{10}$$

For any $\eta \in (0, B_\eta)$, let $\tau = \sqrt{1-\eta}$, and

$$R'_X := \left\{x \in U_X : \Pr_{h\sim\mathcal{T}}\{h(x) = f(x)\} < \tau\right\}. \tag{11}$$

Then we have

$$|R'_X \cap (U_X \setminus W_X)| \leq \frac{\nu}{1-\tau}|U_X \setminus W_X|, \text{ and}$$

$$(1-\frac{\gamma}{\tau})|R_X| \leq |R_X \cap R'_X|, G_X \subseteq R'_X, |R'_X \cap B_X| \geq \frac{\sigma^2-\eta}{1-\eta}|B_X|.$$

Proof. Consider $x \in U_X - W_X$. We have

$$\Pr_{(x,y_x)\sim U}\left\{\Pr_{h\sim\mathcal{T}}\{h(x)=f(x)\}<\tau | f(x)=y_x\right\} \tag{12}$$

$$=\Pr_{(x,y_x)\sim U}\left\{\Pr_{h\sim\mathcal{T}}\{h(x)\neq y_x\}\geq 1-\tau | f(x)=y_x\right\} \tag{13}$$

$$\leq \frac{\nu}{1-\tau}. \tag{14}$$

So only $\frac{\nu}{1-\tau}$ fraction of the data points in $U_X \setminus W_X$ will be put into R_X, proving the first statement.

Now, consider $x \in W_X$. For $x \in R_X$, we have

$$\Pr_{x\sim R_X}\left\{\Pr_{h\sim\mathcal{T}}\{h(x)=f(x)\}\geq\tau\right\}\leq\frac{\gamma}{\tau}. \tag{15}$$

so more than $1-\frac{\gamma}{\tau}$ fraction of the data points in R_X will be put into R'_X.

For $x \in G_X$, since $\eta < 1-\nu^2$, we have $\nu < \tau$. Note that $f(x) \neq y_x$, thus x will be put into R'_X. In the following we consider $x \in B_X = W_X \setminus (R_X \cup G_X)$.

We first show that a significant fraction of B_X has variance larger than η. Since $\sigma_x^2 \in [0,1]$,

$$\sigma^2 = \mathbb{E}[\sigma_x^2 | x \in B_X] \tag{16}$$

$$\leq \Pr\{\sigma_x^2 \leq \eta | x \in B_X\}\cdot\eta + \Pr\{\sigma_x^2 > \eta | x \in B_X\}\cdot 1 \tag{17}$$

$$= (1-\Pr\{\sigma_x^2 > \eta | x \in B_X\})\cdot\eta + \Pr\{\sigma_x^2 > \eta | x \in B_X\} \tag{18}$$

leading to

$$\Pr\{\sigma_x^2 > \eta | x \in B_X\} \geq \frac{\sigma^2-\eta}{1-\eta}. \tag{19}$$

Now it is sufficient to show that any $x \in W_X$ with $\sigma_x^2 > \eta$ will have a small agreement rate $\Pr_{h\sim\mathcal{T}}\{h(x)=f(x)\}$.

$$\Pr_{h\sim\mathcal{T}}[h(x)=f(x)] \leq \sqrt{\sum_{y\in\mathcal{Y}}(\Pr_{h\sim\mathcal{T}}[h(x)=y])^2} \tag{20}$$

$$= \sqrt{\Pr_{h_1,h_2\sim\mathcal{T}}[h_1(x)=h_2(x)]} \tag{21}$$

$$= \sqrt{1-\sigma_x^2} \tag{22}$$

$$< \tau = \sqrt{1-\eta}. \tag{23}$$

So any point $x \in B_X$ with $\sigma_x^2 > \eta$ will fall into R'_X, completing the proof. □

Using the above lemma, we arrive at our main theorem.

Theorem 2 (Restatement of Theorem 1)**.** *Assume in each iteration of the framework, $\tau = \sqrt{1-\eta}$ for some $\eta \in (0, 3B_\eta/4)$ where $B_\eta := \min\{\sigma^2, 1-\nu^2\}$. Let $\sigma_L^2 > 0$ be a lower bound on the diversity σ^2, $\tilde{\gamma}$ be an upper bound on γ, and $\tilde{\nu}$ be an upper bound on ν over all iterations. Then for any $\delta \in (0, \sigma_L^2/4)$, after at most $\lceil 1/\delta \rceil$ iterations, we can get $\frac{|U_X \setminus R_X|}{|U_X|}$ approximates the accuracy $\mathrm{acc}(f)$ and R_X approximates the mis-classified points W_X as follows:*

$$\left|\mathrm{acc}(f) - \frac{|U_X \setminus R_X|}{|U_X|}\right| \leq \max\{\frac{\tilde{\nu}}{1-\tau}(1-e_f), \epsilon e_f\},\ \textit{where}\ \epsilon := \frac{\frac{\tilde{\gamma}}{\tau}\left(1+\frac{\tilde{\nu}}{1-\tau}\frac{1-e_f}{e_f}\right)}{\frac{\sigma_L^2}{4}-\delta+\frac{\tilde{\gamma}}{\tau}}, \tag{24}$$

$$|W_X \triangle R_X| \leq \frac{\tilde{\nu}}{1-\tau}|U_X \setminus W_X| + \epsilon|W_X|. \tag{25}$$

Proof. For each iteration, Lemma 1 implies that the new constructed set R'_X satisfies:

$$|R'_X \cap (U_X \setminus W_X)| \le \frac{\nu}{1-\tau}|U_X \setminus W_X|, \text{ and}$$
$$(1-\frac{\gamma}{\tau})|R_X| \le |R_X \cap R'_X|, G_X \subseteq R'_X, |R'_X \cap B_X| \ge \frac{\sigma^2-\eta}{1-\eta}|B_X|.$$

Since $\eta \le 3\sigma^2/4$,

$$\frac{\sigma^2-\eta}{1-\eta} \ge \frac{\sigma^2}{4}. \tag{26}$$

Therefore,

$$|R'_X \cap B_X| \ge \frac{\sigma^2}{4}|B_X|. \tag{27}$$

Suppose t^* is the first iteration when less than ϵ fraction of W_X is outside R_X. Then in any iteration before t^*, the newly constructed pseudo-labeled set R'_X loses at most $\frac{\tilde{\gamma}}{\tau}$ fraction of R_X, and obtains at least $\frac{\sigma_L^2}{4}$ fraction of $B_X \cap G_X = W_X \setminus R_X$. Since more than ϵ fraction of W_X is outside R_X, it can then be verified that

$$\frac{\sigma_L^2}{4}|W_X \setminus R_X| - \frac{\tilde{\gamma}}{\tau}|R_X| > \delta|W_X \setminus R_X|. \tag{28}$$

Therefore, after each iteration, the framework adds more than δ fraction of $|W_X \setminus R_X|$ in R_X. This can happen at most $1/\delta$ iterations, so $t^* \le \lceil 1/\delta \rceil$.

Now consider the last iteration, and apply Lemma 1, then

$$|R_X \setminus W_X| = |R_X \cap (U_X \setminus W_X)| \le \frac{\nu}{1-\tau}|U_X \setminus W_X| \le \frac{\tilde{\nu}}{1-\tau}|U_X \setminus W_X|. \tag{29}$$

Equation (29) together with the fact that there are less than ϵ fraction of W_X outside R_X lead to the two statements in the theorem. □

By setting the $\eta = 7/16$ and $\delta = 4\tilde{\gamma}/3$, we have the following corollary as an example.

Corollary 1. *Assume in each iteration of the framework, $\nu < 1/2, \sigma^2 > 7/12$, and $\tau = 3/4$ where $B_\eta := \min\{\sigma^2, 1-\nu^2\}$. Let $\sigma_L^2 > 0$ be a lower bound on the diversity σ^2, $\tilde{\gamma}$ be an upper bound on γ, and $\tilde{\nu}$ be an upper bound on ν over all iterations. If $\sigma_L^2 \ge \frac{16\tilde{\gamma}}{3}$, then after at most $\lceil 3/(4\tilde{\gamma}) \rceil$ iterations, we can get $\frac{|U_X \setminus R_X|}{|U_X|}$ approximates the accuracy $\mathrm{acc}(f)$ and R_X approximates the mis-classified points W_X as follows:*

$$\left|\mathrm{acc}(f) - \frac{|U_X \setminus R_X|}{|U_X|}\right| \le \max\{4\tilde{\nu}(1-e_f), \epsilon e_f\}, \text{ where } \epsilon := \frac{16\tilde{\gamma}}{3\sigma_L^2}\left(1 + 4\tilde{\nu}\frac{1-e_f}{e_f}\right), \tag{30}$$
$$|W_X \triangle R_X| \le 4\tilde{\nu}|U_X \setminus W_X| + \epsilon|W_X|. \tag{31}$$

Proof. In Theorem 1, note that $\eta = 7/16$ leads to $\tau = 3/4$. The bounds on ν, γ, and σ^2 comes from the requirement that $3B_\eta/4 > 7/16$ so that there exists such an $\eta \in (7/16, 3B_\eta/4)$. □

B.2 Discussion on Using Ensembles

Our analysis of the framework clearly relies on the effect of self-training. It also shows the benefit of the ensemble: the diversity (combined with low errors of the ensemble on points correctly classified by f) allows to identify mis-classified points.

Here we present more discussion on the approach of using ensembles to estimate the accuracy and to provide further insight into their benefit compared to some other existing approaches. For simplicity, we analyze binary classification with $\mathcal{Y} = \{0, 1\}$ in this section unless stated otherwise. We provide an exact characterization of the estimation error (i.e., how far the agreement rate is from the actual accuracy). It implies that to get a good estimation, one should use ensembles with small prediction

errors on the test points. More importantly, the estimation can be further improved if the ensemble's prediction has proper correlation with f, which then shows the advantage of an ensemble of models instead of a single model.

Let $e_{h,x}$ be the indicator that h mis-classifies x, $e_{\mathcal{T}}$ be the expected error of h on U (over the distribution of h), and e_f be the error of f:

$$e_{h,x} := \mathbb{I}[h(x) \neq y_x], \quad e_{\mathcal{T}} := \mathbb{E}_{h,x}[e_{h,x}], \quad e_f := \mathbb{E}_x[e_{f,x}].$$

Let $\mathrm{ar}_x(f, \mathcal{T})$ be the agreement rate between f and h's on a point x, and $\mathrm{ar}(f, \mathcal{T})$ be that on the whole test set:

$$\mathrm{ar}_x(f, \mathcal{T}) := \Pr_{h \sim \mathcal{T}}\{h(x) = f(x)\},$$
$$\mathrm{ar}(f, \mathcal{T}) := \mathbb{E}_{x \in U_X}[\mathrm{ar}_x(f, \mathcal{T})].$$

Recall that we are using $\mathrm{ar}(f, \mathcal{T})$ as an estimate of the accuracy of f on U.

Lemma 2. *For binary classification,*

$$\mathrm{acc}(f) - \mathrm{ar}(f, \mathcal{T}) = e_{\mathcal{T}}(1 - 2e_f) - 2\mathrm{Cov}(e_{f,x}, e_{h,x}) \tag{32}$$

where $\mathrm{Cov}(e_{f,x}, e_{h,x})$ *is the covariance between* $e_{f,x}$ *and* $e_{h,x}$*. For multi-class classification,*

$$e_{\mathcal{T}}(1 - 2e_f) - 2\mathrm{Cov}(e_{f,x}, e_{h,x}) \leq \mathrm{acc}(f) - \mathrm{ar}(f, \mathcal{T}) \leq e_{\mathcal{T}}(1 - e_f) - \mathrm{Cov}(e_{f,x}, e_{h,x}). \tag{33}$$

Proof. We have

$$\begin{aligned}
\mathrm{acc}(f) - \mathrm{ar}(f, \mathcal{T}) &= \mathbb{E}\{\mathbb{I}[f(x) = y_x]\} - \mathbb{E}\{\mathbb{I}[f(x) = h(x)]\} && (34)\\
&= \mathbb{E}\left\{\mathbb{I}[f(x) = y_x] - \mathbb{I}[f(x) = h(x)]\right\} && (35)\\
&= \mathbb{E}\left\{\mathbb{I}[f(x) \neq h(x)] - \mathbb{I}[f(x) \neq y_x]\right\}. && (36)
\end{aligned}$$

The first term can be decomposed into two parts:

$$\mathbb{I}[f(x) \neq h(x)] = \mathbb{I}[f(x) \neq h(x), f(x) = y_x] + \mathbb{I}[f(x) \neq h(x), f(x) \neq y_x] \tag{37}$$

and the two parts can be transformed as:

$$\begin{aligned}
\mathbb{I}[f(x) \neq h(x), f(x) = y_x] &= \mathbb{I}[h(x) \neq y_x, f(x) = y_x] && (38)\\
&= e_{h,x}(1 - e_{f,x}), && (39)\\
\mathbb{I}[f(x) \neq h(x), f(x) \neq y_x] &= \mathbb{I}[h(x) = y_x, f(x) \neq y_x] && (40)\\
&= e_{f,x}(1 - e_{h,x}) && (41)
\end{aligned}$$

where the second to last line follows from that in binary classification, $f(x) \neq h(x)$ and $f(x) \neq y_x$ is equivalent to $h(x) = y_x$ and $f(x) \neq y_x$. Therefore,

$$\begin{aligned}
\mathrm{acc}(f) - \mathrm{ar}(f, \mathcal{T}) &= e_{\mathcal{T}} - 2\mathbb{E}[e_{h,x}e_{f,x}] && (42)\\
&= e_{\mathcal{T}} - 2(e_{\mathcal{T}}e_f + \mathrm{Cov}(e_{h,x}, e_{f,x})). && (43)
\end{aligned}$$

Rearranging the terms completes the proof.

For multi-class, we can replace (40) by the bounds:

$$\mathbb{I}[h(x) = y_x, f(x) \neq y_x] \leq \mathbb{I}[f(x) \neq h(x), f(x) \neq y_x] \leq \mathbb{I}[f(x) \neq y_x]. \tag{44}$$

□

The bound suggests using $\mathcal{T}$ with a small prediction error $e_{\mathcal{T}}$. More importantly, the estimation can be improved by a proper correlation between f and the ensemble models: even when the ensemble models don't have very small error $e_{\mathcal{T}}$, they can still lead to a good estimation, as long as they have a proper covariance with f. More precisely, typically $e_f < 1/2$, so the covariance should not be negative, but also should not be too positive. For example, when the ensemble models overly agrees with f (e.g., in the extreme case $h(x) = f(x)$ for all $x \in U_X$ and all $h \sim \mathcal{T}$), it leads to over-estimation of the accuracy, and we should decrease the correlation (more discussion in the next subsection).

It is also instructive to compare our method to some existing methods. (1) Our analysis is more general and tighter than that for using a single model in [4]. The setting is a special case of ours. More important, our bound is tighter and reveals that an ensemble with proper correlation can improve the estimation, justifying the advantage of an ensemble over a single model. (2) Our analysis is also more general than the classic notion of calibration. We show that if the ensemble has perfect calibration then the agreement rate equals the accuracy of the pre-trained model. On the other hand, our lemma shows that even without calibration, proper ensembles can still give good estimation.

Detailed comparisons are presented below.

Comparison with Proxy Risk. Recall that the proxy risk method [4] is to use invariant representation domain adaptation methods to find the h of maximum disagreement with f on U_X. That is, it aims to get the $h \in \mathcal{H}$ such that $\mathrm{ar}(f,h)$ is smallest where $\mathcal{H}$ is the set of hypotheses with small errors on the original training data and small distances between the distributions of the representations of the training and test data, i.e.,

$$\mathcal{H} = \{h \in \mathcal{P} : \text{ error of } h \text{ on the training data } + \alpha d(p_S^\phi, p_T^\phi) \leq \epsilon\} \tag{45}$$

where $\mathcal{P}$ is the set of networks for domain adaptation, $d(p_S^\phi, p_T^\phi)$ is some distance between the distributions of the representations of the training and the test data, and α, ϵ are hyperparameters.

The main idea behind the proxy risk method is Lemma 4 in their paper, which states (rephrased to our context):

$$\left| \sup_{h \in \mathcal{H}} \mathbb{E}_{x \sim U_X} \mathbb{I}[f(x) \neq h(x)] - e_f \right| \leq \sup_{h \in \mathcal{H}} e_h \tag{46}$$

where e_h is the error of h on the test set, i.e., $e_h = \mathbb{E}_{(x,y_x)\sim U}\{\mathbb{I}[h(x) \neq y_x]\}$.

Our bound is more general and tighter. We first show that their bound can be recovered from ours. More precisely, the proxy risk method is equivalent to using an ensemble method $\mathcal{T}$ that outputs the $\hat{h} \in \mathcal{H}$ of maximum disagreement with f. Then the output distribution of $\mathcal{T}$ concentrates on $\hat{h} \in \mathcal{H}$. Our bound then leads to:

$$\begin{aligned}
\left| \sup_{h \in \mathcal{H}} \mathbb{E}_x \mathbb{I}[f(x) \neq h(x)] - e_f \right| &= \left| \mathbb{E}_x \mathbb{I}[f(x) \neq \hat{h}(x)] - e_f \right| && (47)\\
&= \left| \mathrm{acc}(f) - \mathbb{E}_{h\sim\mathcal{T}}[\mathrm{ar}(f,h)] \right| && (48)\\
&= |e_{\mathcal{T}} - 2\mathbb{E}[e_{f,x} e_{h,x}]| && (49)\\
&= |e_{\hat{h}} - 2\mathbb{E}_x[e_{f,x} e_{h,x}]|. && (50)
\end{aligned}$$

Since $e_{f,x}$ and $e_{h,x}$ are in $\{0,1\}$, it is easy to see that

$$0 \leq \mathbb{E}_x[e_{f,x} e_{h,x}] \leq \min\{\mathbb{E}_x[e_{f,x}], \mathbb{E}_x[e_{\hat{h},x}]\} = \min\{e_f, e_{\hat{h}}\}. \tag{51}$$

Therefore,

$$\begin{aligned}
\left| \sup_{h \in \mathcal{H}} \mathbb{E}_x \mathbb{I}[f(x) \neq h(x)] - e_f \right| &= |e_{\hat{h}} - 2\mathbb{E}_x[e_{f,x} e_{h,x}]| && (52)\\
&\leq e_{\hat{h}} && (53)\\
&\leq \sup_{h \in \mathcal{H}} e_h && (54)
\end{aligned}$$

recovering the bound in the proxy risk paper.

The above calculation also shows that our bound is tighter. Their bound is only for the case when only one check model $\hat{h}$ is learned and also for the worst case. First, it is challenging to find an $\hat{h}$ with a small error in many practical scenarios. For example, the test data contains outlier inputs which are not similar to the training data. It is then unlikely to find an $\hat{h}$ with small errors on these data points since no enough label information is available. However, it is still possible to have a good estimation of the accuracy, since the outlier data are different from the training data and thus can be detected, and we know that f is likely to make errors on them. Second, the bound is also too pessimistic. In the experiments, we observed that the proxy risk method can still achieve reasonable

estimation (about 10% away from the true accuracy), even when the error of $\hat{h}$ is very large (e.g., $> 60\%$ while $\sup_{h\in\mathcal{H}} e_h$ is even larger).

Our lemma suggests that by allowing an ensemble $h \sim \mathcal{T}$ with proper diversity, we have more flexibility and can significantly improve the pessimistic bound. For the example given above, the ensemble method can potentially handle the outlier input data: for hypotheses agreeing with the training data, they are still likely to have disagreement on the outlier data and this disagreement thus reveals the potential error of f there, leading to an accurate estimation of the accuracy. We thus propose to use an ensemble method for estimating the accuracy.

Comparison with Calibration. A classic notion for uncertainty estimation is calibration of the machine learning model. It is well-known that if the pre-trained model f outputs confidence scores for class labels and the confidence is well-calibrated, then the average confidence approximates its accuracy. Unfortunately, it has been observed that many machine learning systems, in particular modern neural networks, are poorly calibrated, especially on test data with distribution shift [14, 30] which is the most interesting case for accuracy estimation.

On the other hand, one can hope to obtain an ensemble of models that is well calibrated, such as the deep ensemble method [25]. Below we show that well-calibration of the ensemble implies the agreement rate between the ensemble and the pre-trained model is a good estimation of the accuracy of the pre-trained model. Formally, we consider the simplified setting of perfect calibration defined as follows.

Definition 4 (Perfect Calibration)**.** *An ensemble $\mathcal{T}$ of classifiers has perfect calibration on the dataset $U = \{(x, y_x)\}$, if for any class label $k \in \mathcal{Y}$ and any $p \in [0, 1]$,*

$$\Pr_{(x,y_x)\sim U}[y_x = k | C_k(x) = p] = p. \tag{55}$$

where $C_k(x) := \Pr_{h\sim\mathcal{T}}[h(x) = k]$ is the confidence score of $\mathcal{T}$ for label k on the input x.

(The definition and the later analysis also applies to a classifier $C_k(x)$ outputting confidence scores, or replacing U with a data distribution.)

Proposition 1. *If the ensemble $\mathcal{T}$ has perfect calibration, then* $\mathrm{ar}(f, \mathcal{T}) = \mathrm{acc}(f)$.

Proof. By definition, we have

$$\begin{aligned}
\mathrm{ar}(f, \mathcal{T}) &= \Pr_{h,x}[h(x) = f(x)] && (56)\\
&= \mathbb{E}_x \left[\Pr_h[h(x) = f(x)]\right] && (57)\\
&= \mathbb{E}_x \left[C_{f(x)}(x)\right] && (58)\\
&= \mathbb{E}\Big\{ \mathbb{E}\left\{\mathbb{E}\left[C_{f(x)}(x) | C_k(x) = p\right] | f(x) = k\right\} \Big\} && (59)\\
&= \mathbb{E}\Big\{ \mathbb{E}\left\{\mathbb{E}\left[C_k(x) | C_k(x) = p\right] | f(x) = k\right\} \Big\} && (60)\\
&= \mathbb{E}\Big\{ \mathbb{E}\left\{\mathbb{E}\left[p | C_k(x) = p\right] | f(x) = k\right\} \Big\} && (61)\\
&= \mathbb{E}\Big\{ \mathbb{E}\left\{\mathbb{E}\left[y_x = k | C_k(x) = p\right] | f(x) = k\right\} \Big\} && (62)\\
&= \mathbb{E}[y_x = f(x)] && (63)\\
&= \mathrm{acc}(f). && (64)
\end{aligned}$$

The third line follows from the definition of $C_k(x)$, the fourth line from the law of total expectation, the seventh line from perfect calibration, and the eighth line from the law of total expectation. □

On the other hand, our Lemma 2 is more general: it shows even if the ensemble is not well calibrated, it is still possible for the agreement rate to be a good estimation of the accuracy. For illustration, consider a simple example with 4 points in U, and only one model h from $\mathcal{T}$ (if $h(x) = k$, we view

x	x_2^-	x_1^-	x_1^+	x_2^+
y_x	$-$	$-$	$+$	$+$
$f(x)$	$-$	$+$	$-$	$+$
$h(x)$	$-$	$+$	$+$	$-$

Table 2: An illustrative example showing even if the ensemble is not well calibrated, it is still possible for the agreement rate to be a good estimation of the accuracy.

it as $\Pr_h[h(x) = k] = 1$). The predictions of f and h are shown in Table 2. It is easy to see that h is not well-calibrated, e.g.,

$$\Pr_h[y_x = +|C_+(x) = 1] = \Pr_h[y_x = +|h(x) = +] = 1/2 \ll 1.$$

On the other hand, $\mathrm{ar}(f, \mathcal{T}) = \mathrm{acc}(f) = 1/2$. From the perspective of Lemma 2, although the ensemble has a large error $e_{\mathcal{T}} = 1/2$, its predictions and those of f are properly correlated, such that

$$\mathrm{acc}(f) - \mathrm{ar}(f, \mathcal{T}) = e_{\mathcal{T}} - 2\mathbb{E}[e_{h,x}e_{f,x}] = \frac{1}{2} - 2 \cdot \frac{1}{4} = 0$$

leading to an accurate estimation of the accuracy of f.

C Experimental Details

C.1 Setup

C.1.1 Computing Infrastructure

We run all experiments with PyTorch and NVIDIA GeForce RTX 2080Ti GPUs.

C.1.2 Dataset

In our problem setting, we need a training dataset D and a test dataset U. We evaluate our methods on five dataset categories. Each dataset category contains several evaluation tasks (or several training-test dataset pairs). We introduce each of the dataset categories below.

Digits. We investigate four digit datasets, which are MNIST [26], MNIST-M [12], SVHN [29] and USPS [19]. They all contain digit images with digits from 0 to 9. MNIST contains 60,000 training images and 10,000 test images; MNIST-M contains 59,001 training images and 9,001 test images; SVHN contains 73,257 training images and 26,032 test images; USPS contains 7,291 training images and 2,007 test images. We can construct 12 different training-test dataset pairs from them.

Office-31. [33] is the most widely used dataset for visual domain adaptation, with 4,652 images and 31 categories collected from three distinct domains: *Amazon*(A), *Webcam*(W), and *Dslr*(D). *Amazon* contains 2,817 images; *Webcam* contains 795 images; *Dslr* contains 498 images. The images are cropped to be the size of 224×224. We can construct 6 different training-test dataset pairs from them.

CIFAR10-C. We use CIFAR10 [24] as the training dataset and CIFAR10-C [16] as the test dataset. CIFAR10 contains 50,000 training images and 10,000 test images. CIFAR10-C contains test images with 19 corruption types and 5 severity levels. We only consider severity level of 5. For each corruption type, it contains 10,000 test images generated from CIFAR10 test images by applying the corruption. We have 19 different training-test dataset pairs in total.

iWildCam. The iWildCam 2020 Competition Dataset [1] contains animal images with 186 species collected by various camera traps. The task is multi-class species classification. We use a variant of it, which is proposed by [23]. It contains 142,202 training images and 7,861 in-distribution test images from 245 locations. The validation set contains 20,784 images from 32 locations and the test set contains 38,943 images from 47 locations. The locations in the validation set and the test set are different from those in the training set. We create 10 target datasets by sampling data from the validation set and the test set. Each target dataset contains images from 50 locations, which

are different from those in the training data. So we have 10 training-test dataset pairs. To reduce computational cost, we resize each image to 64×64.

Amazon Review. The Multi-Domain Sentiment Dataset constructed in [2] is a dataset for sentiment domain adaptation. It contains Amazon product reviews for four different product types: books, DVDs, electronics and kitchen appliances. For each domain, it has 1,000 positive and 1,000 negative examples. We name it Amazon Review dataset and construct 12 different training-test dataset pairs from it.

C.1.3 Implementation and Hyperparameters

In our algorithms, we need to set hyperparameters T, N, α and γ. We specify α in the training configuration (See Section C.1.4). We set $T = 5$ for all dataset pairs. We set $N = 20$ for Amazon Review dataset and set $N = 5$ for other dataset categories. For the implementation of self-training objective, we randomly sample data points from $D \cup R$ in batches and weight the loss term on the pseudo labeled data by γ. We set $\gamma = 0.1$ for all dataset pairs. For dataset pairs in Digits that use USPS as target dataset, we repeat R ten times and then add it to the training set due to the much smaller size of the test set in USPS compared to the source training set. This is equivalent to multiplying γ by 10.

C.1.4 Model Architecture and Training Configuration

We introduce the model architectures and the training configurations for each dataset category below.

Digits. We use a neural network named CNN-BN, which has two convolutional layers, two full connected layers and batch normalization layers as the typical DNN model. The DANN architecture is adapted from the one used in [4]. We train all models using Adam optimizer with learning rate of 10^{-3} and batch size of 128. For supervised learning, we train the model for 20 epochs. For domain adaptive learning, we train DANN for 100 epochs. We adopt the original progressive training strategy for the discriminator [12] where the weight α for the domain-invariant loss is initiated at 0 and is gradually changed to 0.1 using the schedule $\alpha = (\frac{2}{1+\exp(-10 \cdot p)} - 1) \cdot 0.1$, where p is the training progress linearly changing from 0 to 1. When fine-tuning the models, we set $\alpha = 0.1$ and use Adam optimizer with learning rate of 10^{-3}. The model architecture for DANN is presented below.

Encoder
nn.Conv2d(3, 64, kernel_size=5)
nn.BatchNorm2d
nn.MaxPool2d(2)
nn.ReLU
nn.Conv2d(64, 128, kernel_size=5)
nn.BatchNorm2d
nn.Dropout2d
nn.MaxPool2d(2)
nn.ReLU
nn.Conv2d(128, 128, kernel_size=3, padding=1)
nn.BatchNorm2d
nn.ReLU
$\times 2$

Predictor
nn.Conv2d(128, 128, kernel_size=3, padding=1)
nn.BatchNorm2d
nn.ReLU
$\times 3$
flatten
nn.Linear(2048, 256)
nn.BatchNorm1d
nn.ReLU
nn.Linear(256, 10)
nn.Softmax

Discriminator
nn.Conv2d(128, 128, kernel_size=3, padding=1)
nn.ReLU
$\times 5$
Flatten
nn.Linear(2048, 256)
nn.ReLU
nn.Linear(256, 2)
nn.Softmax

Office-31. We use ResNet50 [15] as the typical DNN model. For the DANN architecture, we use ResNet50 followed by a four-layer fully connected network with width 256 as the feature extractor. The main classifier is a three-layer fully connected network with width 256 and the auxiliary classifier is a seven-layer fully connected network with width 256. We train all models for 100 epochs using Adam optimizer with batch size of 32 and learning rate schedule. The initial learning rate is 10^{-5} and it decreases to 10^{-6} after 50 epochs training. We augment the training data using random resized crop and random horizontal flip. The weight α for DANN training is initiated at 0 and is gradually changed to 1 using the same schedule discussed above. When fine-tuning the models, we set $\alpha = 1$ and use Adam optimizer with learning rate of 5×10^{-6}. The model architecture for DANN is presented below.

Encoder
ResNet50(pretrained=True)
nn.Linear(2048, 256)
nn.ReLU
nn.Linear(256, 256)
nn.ReLU
$\times 4$

Predictor
nn.Linear(256, 256)
nn.BatchNorm1d
nn.ReLU
$\times 2$
nn.Linear(256, 31)
nn.Softmax

Discriminator
nn.Linear(256, 256)
nn.ReLU
$\times 6$
nn.Linear(256, 2)
nn.Softmax

CIFAR10-C. We use ResNet34 [15] as the typical DNN model. For the DANN architecture, we use ResNet34 as the feature extractor. The main classifier is a three-layer fully connected network with width 256 and the auxiliary classifier is a seven-layer fully connected network with width 256. We train all models for 100 epochs using Stochastic Gradient Decent (SGD) optimizer with Nesterov momentum and learning rate schedule. We set momentum 0.9 and ℓ_2 weight decay with a coefficient of 10^{-4}. The initial learning rate is 0.1 and it decreases by 0.1 at 50, 75 and 90 epoch respectively. The batch size is 128. We augment the training data using random crop with padding and random horizontal flip. The weight α for DANN training is initiated at 0 and is gradually changed to 0.1 using the same schedule discussed above. When fine-tuning the models, we set $\alpha = 0.1$ and use the same SGD optimizer with a learning rate of 10^{-4}. The model architecture for DANN is presented below.

Encoder
ResNet34
nn.Linear(512, 256)
nn.ReLU

Predictor
nn.Linear(256, 256)
nn.ReLU
$\times 2$
nn.Linear(256, 10)
nn.Softmax

Discriminator
nn.Linear(256, 256)
nn.ReLU
$\times 6$
nn.Linear(256, 2)
nn.Softmax

iWildCam. We use ResNet50 [15] as the typical DNN model. For the DANN architecture, we use ResNet50 followed by a four-layer fully connected network with width 256 as the feature extractor. The main classifier is a three-layer fully connected network with width 256 and the auxiliary classifier is a seven-layer fully connected network with width 256. We train all models for 50 epochs using Adam optimizer with batch size of 128 and learning rate of 10^{-5}. The weight α for DANN training is initiated at 0 and is gradually changed to 1 using the same schedule discussed above. When fine-tuning the models, we set $\alpha = 1$ and use Adam optimizer with learning rate of 10^{-5}. The model architecture for DANN is presented below.

Encoder
ResNet50(pretrained=True)
nn.Linear(2048, 256)
nn.ReLU
nn.Linear(256, 256)
nn.ReLU
$\times 4$

Predictor
nn.Linear(256, 256)
nn.BatchNorm1d
nn.ReLU
$\times 2$
nn.Linear(256, 186)
nn.Softmax

Discriminator
nn.Linear(256, 256)
nn.ReLU
$\times 6$
nn.Linear(256, 2)
nn.Softmax

Amazon Review. We use TF-IDF [34] to transform texts into feature vectors with dimension 5,000. Then we build fully connected networks with ReLU activation. For typical DNN model, we use a four-layer fully connected network with width 128. For the DANN architecture, we use a four-layer fully connected network with width 128 as the feature extractor. The main classifier is a three-layer fully connected network with width 128 and the auxiliary classifier is a seven-layer fully connected network with width 256. We train all models for 50 epochs using Adam optimizer with batch size of 8 and learning rate of 10^{-3}. The weight α for DANN training is initiated at 0 and is gradually changed to 1 using the same schedule discussed above. When fine-tuning the models, we set $\alpha = 1$ and use Adam optimizer with learning rate of 10^{-3}. The model architecture for DANN is presented below.

Encoder
nn.Linear(5000, 128)
nn.ReLU
nn.Linear(128, 128)
nn.ReLU
$\times 3$

Predictor
nn.Linear(128, 128)
nn.ReLU
$\times 2$
nn.Linear(128, 2)
nn.Softmax

Discriminator
nn.Linear(128, 256)
nn.ReLU
nn.Linear(256, 256)
nn.ReLU
$\times 5$
nn.Linear(256, 2)
nn.Softmax

C.1.5 Evaluation Metrics

Unsupervised Accuracy Estimation. We use the absolute estimation error $|\hat{\text{acc}} - \text{acc}(f, U)|$ to measure the performance of the accuracy estimator $\hat{\text{acc}}$.

Error Detection. We formulate error detection as a binary classification problem (a test point detected to be mis-classified is regarded as in the positive class). Then we can use F1 score to measure the performance of the algorithms quantitatively.

C.1.6 Baselines

We consider the following baselines:

Proxy Risk. We consider proxy risk method [4] as a baseline for both unsupervised accuracy estimation and error detection tasks. In proxy risk method, they train the check model and fine-tune it to maximize the disagreement using a separate target training dataset sampled from the distribution of the test dataset U_X. However, in practice, we might not have such a training dataset. Thus, in our implementation, we will use U_X as the training dataset if a separate target training dataset is not available. Specifically, for Digits, we use a separate target training dataset, while for other dataset categories, we use U_X as the target training dataset. Note that our method doesn't need a separate target training dataset. We follow their settings to set the hyperparameters. Specifically, for all experiments, we set $\lambda = 50$ and $T_2 = 20$. In Digits and CIFAR10-C experiments, we set $\alpha = 0.1$ and $\epsilon = 1.1 \cdot \epsilon_0$; in Office-31 and iWildCam experiments, we set $\alpha = 1$ and $\epsilon = 1.05 \cdot \epsilon_0$, where ϵ_0 is the value of divergence loss $R_S(f'g') + \alpha d(p_S^{g'}(Z), p_T^{g'}(Z))$ computed using the pre-trained check model $h' = f'g'$. The way to pre-train the check model h' is the same as the DANN training method in Section C.1.4. For fine-tuning the check model to maximize the disagreement, we use SGD optimizer for CIFAR10-C and Adam optimizer for other datasets. The learning rate is set to be 10^{-3} in Digits, 10^{-6} in Office-31, 10^{-4} in CIFAR10-C, 10^{-5} in iWildCam and 10^{-3} in Amazon Review.

Avg Conf. We consider the average confidence score as a baseline for unsupervised accuracy estimation task. We know if the model is well calibrated on U_X, then the average confidence of the model on U_X can represent the accuracy. [9] propose to use the average confidence score as a metric to measure the performance drop of a model. In our context, it is equivalent to using the average confidence as a measure for the accuracy.

Ens Avg Conf. Deep ensemble [25] has been shown to be an effective technique to improve the model calibration. So we consider the average confidence score of an ensemble model as a baseline for unsupervised accuracy estimation task. We use the same architecture and training procedure of the pre-trained model f to train the models in the ensemble. The number of models in the ensemble is 10.

MSP. We consider Maximum Softmax Probability (MSP) [17] as a baseline for error detection. We pick the confidence threshold using a test dataset D^{test} sampled from the training distribution $P_{X,Y}$ such that the fraction of data points in D^{test} whose confidence scores are less than the threshold is equal to the error of the given model on D^{test}.

Trust Score. We consider Trust Score [21] as a baseline for error detection task. We use the logit layer as the input to Trust Score. And we pick the threshold using a test dataset D^{test} sampled from the training distribution $P_{X,Y}$ such that the fraction of data points in D^{test} whose trust scores are less than the threshold is equal to the error of the given model on D^{test}.

C.2 Full Plots for Unsupervised Accuracy Estimation

We plot the results for using typical DNN as the model f's architecture in Figure 3 and the results for using DANN-arch as the model f's architecture in Figure 4.

C.3 Multiple Runs of Experiments

Since proxy risk method and our method require training and fine-tuning the models, there might be some variance in the results. Thus, we repeat each experiment in Table 1 for proxy risk method and our method five

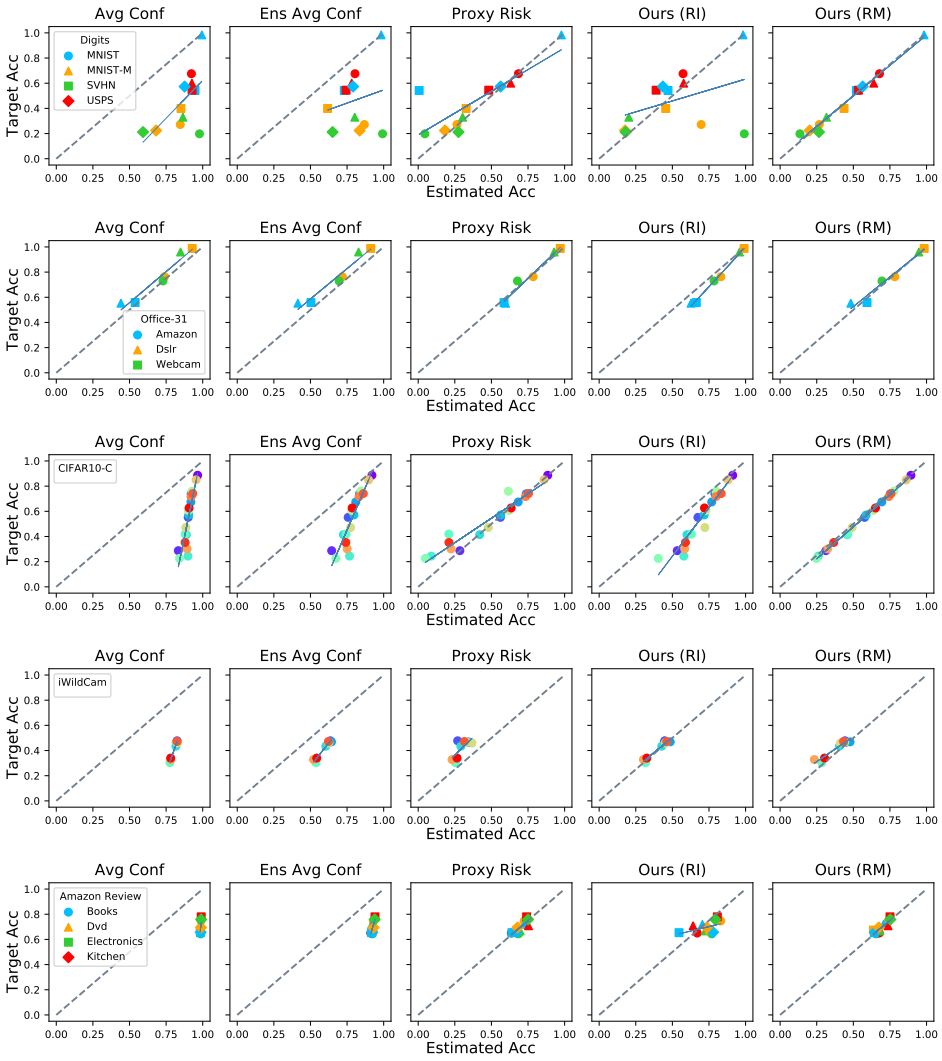


Figure 3: Accuracy estimation results for each dataset pair. We use typical DNN as the architecture for the model f. We use symbols to represent training datasets and colors to represent test datasets. For CIFAR10-C and iWildCam, there is only one training dataset with multiple test datasets. The dashed line represents perfect prediction (target accuracy = estimated accuracy). Points beneath (above) the dashed line indicate overestimation (underestimation). The solid lines are regression lines of the results.

times, and show the box plots of the results measured by average absolute estimation error and average F1 score (See Figure 5). The results show that our method is consistently better than proxy risk method and the variance of the results of our method is generally smaller than that of proxy risk method.

C.4 Validating the Theoretical Analysis

Our analysis relies on three conditions: (A) The ensemble models make small error on the test inputs correctly classified by f (and thus tend to over-estimate the accuracy); (B) The ensemble models mostly disagree with f on R_X; (C) The ensemble models have large diversity on B_X. Since we don't use threshold in our implementation, B_X here is the data points in $W_X \setminus R_X$ where the prediction of the ensemble (via majority vote) is wrong. Our experiments on six dataset pairs over two dataset categories (MNIST→MNIST-M, MNIST→USPS and MNIST-M→USPS on the Digits dataset category; Amazon→Dslr, Dslr→Webcam, and Amazon→Webcam on the Office-31 dataset category) show that empirically they are roughly satisfied.

Table 3 shows the actual accuracy, the estimated accuracy obtained via the initial ensemble (Estimated Acc w/o self-training), and the estimated accuracy after applying the self-training (Estimated Acc w/ self-training). The comparison confirms the over-estimation and shows the self-training can rectify that. The table also shows the upper bound $\tilde{\nu}$ on average error on correct points ν over all iterations is small. The upper bound $\tilde{\gamma}$ on the

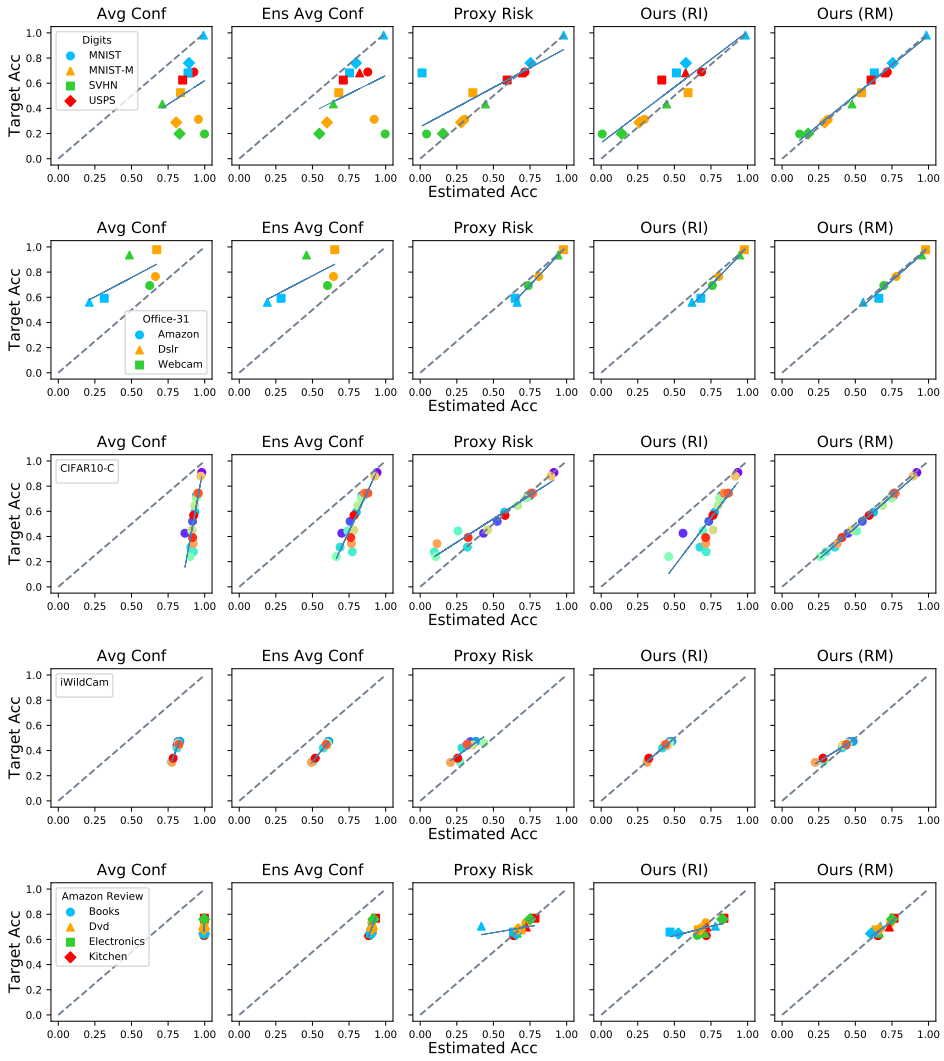
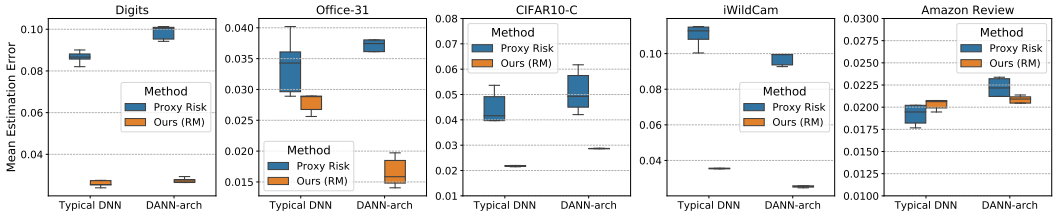


Figure 4: Accuracy estimation results for each dataset pair. We use DANN-arch as the architecture for the model f. We use symbols to represent training datasets and colors to represent test datasets. For CIFAR10-C and iWildCam, there is only one training dataset with multiple test datasets. The dashed line represents perfect prediction (target accuracy = estimated accuracy). Points beneath (above) the dashed line indicate overestimation (underestimation). The solid lines are regression lines of the results.

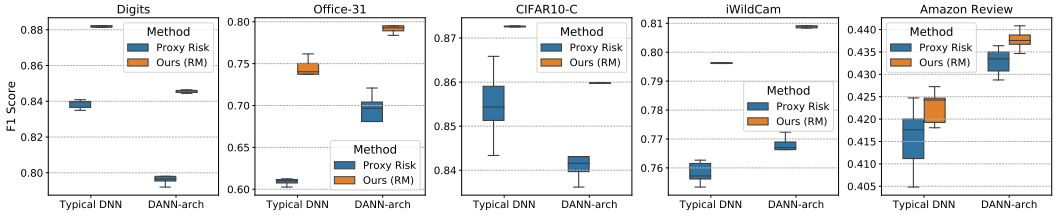
average probability of agreement γ between the ensemble models and f on R_X over all iterations is close to 0, so the ensemble models mostly disagree with f on R_X. Finally, the lower bound σ_L^2 on the diversity of the ensemble σ^2 over all iterations is relatively large.

Besides, to support our claims regarding pseudo-labels, we perform experiments (using the same settings as the experiments in Table 3) on MNIST→MNIST-M, our observations are: (1) the pseudo-labels are not all correct. The accuracies of the pseudo-labels over three iterations are 93.89%, 94.02% and 93.82%. (2) the new ensemble will become more accurate with the help of correct pseudo-labels. The accuracies of the ensembles on U over three iterations are 89.56%, 93.51% and 94.53%. (3) the new ensemble will be less diverse on the pseudo-labeled data. The upper bound of average σ_x^2 on the pseudo-labeled data over all iterations is 4.78%, which is relatively small compared to $\sigma_L^2 = 26.54\%$.

Furthermore, we perform an experiment (using the same settings as the experiments in Table 3) on MNIST→MNIST-M to verify if our conditions still hold when the pre-trained classifier is a majority vote over an ensemble. The ensemble contains 10 models trained on D with different random initializations. The results are similar to those for a single model. Our success conditions are still roughly satisfied and the accuracy estimation is great: $\tilde{\nu} = 2.61\%$, $\tilde{\gamma} = 0.59\%$, $\sigma_L^2 = 28.94\%$ and the estimation error is 0.0009.



(a) Unsupervised Accuracy Estimation



(b) Error Detection

Figure 5: Results for multiple runs of experiments.

Dataset Category	Digits			Office-31		
Dataset Pair	M→MM	M→U	MM→U	A→D	D→W	A→W
Actual Acc	27.19	67.56	60.04	76.31	95.97	72.96
Estimated Acc w/o self-training	33.05	70.30	69.76	80.52	95.85	74.34
Estimated Acc w/ self-training	27.50	68.21	64.28	78.11	95.09	70.57
$\tilde{\nu}$	3.15	2.18	4.22	6.16	2.17	11.11
$\tilde{\gamma}$	0.57	0.90	3.82	1.92	0.20	0.29
σ_L^2	26.54	12.89	24.57	15.61	8.80	14.67

Table 3: Empirical results to support the theoretical analysis. We use typical DNN as the architecture for the model f. M is MNIST, MM is MNIST-M, U is USPS, A is Amazon, D is Dslr and W is Webcam. The ensemble training algorithm we use is $\mathcal{T}_{\text{RM}}$. On Digits, we use $N = 10$ and $T = 3$ while on Office-31, we use $N = 15$ and $T = 2$. All values are percentages.

C.5 Evaluation on Pre-trained Models with Various Architectures

We evaluate our method on pre-trained models with different deep learning model architectures on Digits. The architectures considered are Convolutional Neural Network (CNN) [27], Convolutional Neural Network with Batch Normalization (CNN-BN), ResNet18, ResNet34 [15], DenseNet40 and DenseNet100 [18]. The results in Table 4 demonstrate that our method consistently outperforms other methods on pre-trained models with various architectures.

C.6 Implementation of the Framework with Explicit Thresholding

We also implement the Framework 1 with explicit thresholding. We use $\mathcal{T}_{\text{RM}}$ to construct the ensemble $\{h_i\}_{i=1}^N$ and use the following pseudo-labeling strategy: for each $x \in R_X$, if the majority vote of the ensemble on x is different from $f(x)$, then we use the majority vote as the pseudo-label; otherwise, we use a random label that is different from $f(x)$ as the pseudo-label. We perform experiments on Digits dataset category to compare the method using thresholding with Algorithm 1 where we don't use thresholding. The results in Table 5 show that using explicit thresholding leads to similar performance as Algorithm 1.

C.7 Ablation Study on CIFAR10-C

We perform an additional ablation study on CIFAR10-C and the results are shown in Figure 6. We observe a similar trend as that on Digits.

C.8 Analysis on Target Accuracy

The target accuracy of our ensemble check models can be higher or lower than that of the pre-trained model f depending on the datasets and in both cases, our method can achieve good performance. In Table 6, we

Task	Accuracy Estimation		Error Detection	
Architecture of f	Method	Estimation Error ↓	Method	F1 score ↑
CNN	Avg Conf	0.252±0.159	MSP	0.595±0.228
	Ens Avg Conf	0.141±0.111	Trust Score	0.567±0.125
	Proxy Risk	0.080±0.167	Proxy Risk	0.817±0.129
	Ours (RI)	0.120±0.125	Ours (RI)	0.735±0.170
	Ours (RM)	**0.020**±0.022	Ours (RM)	**0.865**±0.075
CNN-BN	Avg Conf	0.404±0.180	MSP	0.467±0.195
	Ens Avg Conf	0.337±0.229	Trust Score	0.496±0.195
	Proxy Risk	0.085±0.142	Proxy Risk	0.844±0.118
	Ours (RI)	0.164±0.218	Ours (RI)	0.698±0.235
	Ours (RM)	**0.023**±0.020	Ours (RM)	**0.881**±0.084
ResNet18	Avg Conf	0.434±0.330	MSP	0.365±0.193
	Ens Avg Conf	0.245±0.220	Trust Score	0.408±0.176
	Proxy Risk	0.111±0.165	Proxy Risk	0.787±0.178
	Ours (RI)	0.173±0.154	Ours (RI)	0.684±0.246
	Ours (RM)	**0.039**±0.043	Ours (RM)	**0.834**±0.128
ResNet34	Avg Conf	0.399±0.261	MSP	0.484±0.114
	Ens Avg Conf	0.288±0.242	Trust Score	0.551±0.153
	Proxy Risk	0.115±0.196	Proxy Risk	0.795±0.179
	Ours (RI)	0.181±0.170	Ours (RI)	0.670±0.168
	Ours (RM)	**0.042**±0.043	Ours (RM)	**0.834**±0.129
DenseNet40	Avg Conf	0.389±0.309	MSP	0.470±0.097
	Ens Avg Conf	0.199±0.205	Trust Score	0.560±0.064
	Proxy Risk	0.115±0.168	Proxy Risk	0.794±0.185
	Ours (RI)	0.170±0.154	Ours (RI)	0.697±0.239
	Ours (RM)	**0.041**±0.046	Ours (RM)	**0.833**±0.144
DenseNet100	Avg Conf	0.388±0.350	MSP	0.430±0.229
	Ens Avg Conf	0.248±0.254	Trust Score	0.533±0.155
	Proxy Risk	0.127±0.176	Proxy Risk	0.759±0.200
	Ours (RI)	0.186±0.182	Ours (RI)	0.677±0.256
	Ours (RM)	**0.047**±0.052	Ours (RM)	**0.803**±0.160

Table 4: Evaluate the methods for unsupervised accuracy estimation and error detection on pre-trained models with various model architectures. The models are trained using supervised learning method. We show the mean and standard deviation of absolute estimation error and F1 score (mean±std). The numbers are calculated over the training-test dataset pairs constructed in the Digits dataset category. **Bold** numbers are the superior results.

Method		Accuracy Estimation	Error Detection
Thresholding	Threshold τ	Estimation Error ↓	F1 score ↑
Yes	0.5	0.026±0.023	0.881±0.086
Yes	0.4	0.033±0.030	0.877±0.086
No	-	0.023±0.020	0.881±0.084

Table 5: Results for comparing our methods with and without thresholding. We use typical DNN as the architecture for the model f. We show the mean and standard deviation of absolute estimation error and F1 score (mean±std). The numbers are calculated over the training-test dataset pairs in Digits dataset category.

compare the target accuracy of the pre-trained model f and the ensemble check models $\{h_i\}_{i=1}^N$ generated by our method with $\mathcal{T}_{\text{RM}}$ on some dataset pairs in Digits and iWildCam. The results show that on Digits where the domain adaptation method can work well, the target accuracy of $\{h_i\}_{i=1}^N$ built by $\mathcal{T}_{\text{RM}}$ is usually higher than that of the pre-trained model f. In such cases, the high accuracy of $\{h_i\}_{i=1}^N$ plays an important role in the good performance of our method. However, on iWildCam where the domain adaptation method fails, the target accuracy of $\{h_i\}_{i=1}^N$ is typically not higher than that of f. In such cases, our method can still achieve good performance due to the diversity of the ensemble.

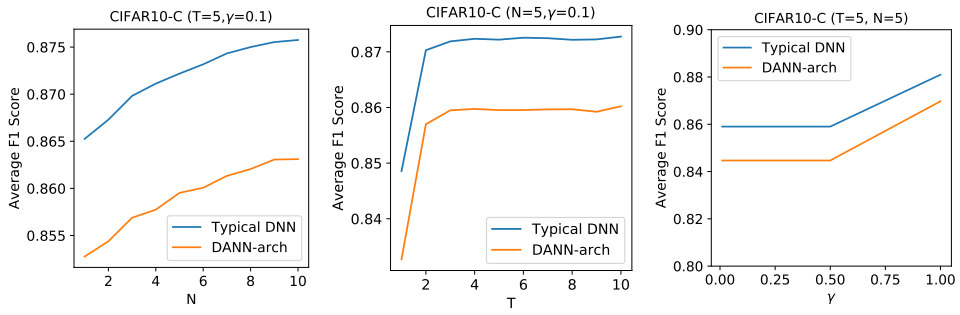


Figure 6: Ablation study for the effect of ensemble and self-training techniques on CIFAR10-C. N is the number of models in the ensemble, T is the number of self-training iterations, and γ is the weighting parameter for the loss term on the pseudo-labeled data. The ensemble training algorithm we use is $\mathcal{T}_{\text{RM}}$.

Dataset Category	Dataset Pair	Target Acc. of f	Ours (RM)		
			Target Acc. of $\{h_i\}_{i=1}^N$	Estimation Error	F1 score
Digits	M→MM	27.19%	93.99%	0.0013	0.9901
	M→U	67.56%	92.53%	0.0065	0.9418
	S→U	54.46%	67.46%	0.0010	0.7941
iWildCam	0→1	46.40%	39.66%	0.0202	0.7872
	0→2	46.90%	42.73%	0.0335	0.7996
	0→9	34.01%	24.50%	0.0376	0.7871

Table 6: Results of comparing the target accuracy of the pre-trained model f and the ensemble check models $\{h_i\}_{i=1}^N$. We use typical DNN as the architecture for the model f. The prediction of the ensemble $\{h_i\}_{i=1}^N$ is produced via majority vote. M is MNIST, MM is MNIST-M, U is USPS and S is SVHN.

C.9 Analysis on Proxy Risk

Proxy Risk has two stages: first train a check model using domain-invariant representations (DIR) and then fine-tune it to maximize the disagreement between the pre-trained model f and the check model h on the target data while maintaining small DIR loss. To study the effect of the disagreement maximization in Proxy Risk, we perform an experiment on Digits for the typical DNN f by removing the disagreement maximization component from Proxy Risk. The results show that without disagreement maximization, the mean F1 score of Proxy Risk on Digits will decrease from 0.844 to 0.812.

Besides, to see whether combining Proxy Risk with ensemble could lead to better results than our method, we perform an experiment on Digits for a variant of Proxy Risk: the check model is an ensemble of models $h'_1, \ldots, h'_t$ via majority vote and each model h'_j in the ensemble is fine-tuned from the pre-trained check model h' using Proxy Risk's maximizing disagreement training objective with different randomness. We set $t = 5$ and use typical DNN as the model architecture of f. The experimental result on Digits is: the mean absolute estimation error is 0.0814 and the mean F1 score is 0.8515. The performance of this variant of Proxy Risk is worse than that of our method with $\mathcal{T}_{\text{RM}}$ (the result for our method is: mean estimation error is 0.0230 and mean F1 score is 0.8810).