ASPEST: BRIDGING THE GAP BETWEEN ACTIVE LEARNING AND SELECTIVE PREDICTION

Jiefeng Chen[1]* **Jinsung Yoon**[2] **Sayna Ebrahimi**[2] **Sercan Arik**[2] **Somesh Jha**[1,2] **Tomas Pfister**[2]
[1] University of Wisconsin-Madison [2] Google
{jiefeng,jha}@cs.wisc.edu
{jinsungyoon,saynae,soarik,tpfister}@google.com

ABSTRACT

Selective prediction aims to learn a reliable model that abstains from making predictions when the model uncertainty is high. These predictions can then be deferred to a humans for further evaluation. In many real-world scenarios, however, the distribution of test data is different from the training data, resulting in more inaccurate predictions, necessitating increased human labeling. Active learning circumvents this by only querying the most informative examples and, in several cases, has been shown to lower the overall labeling effort. We bridge the gap between selective prediction and active learning, proposing a new learning paradigm called *active selective prediction* which learns to query more informative samples from the shifted target domain while increasing accuracy and coverage. We propose a simple but effective solution, ASPEST, that trains ensembles of model snapshots using self-training with their aggregated outputs as pseudo labels. Extensive experiments demonstrate that active selective prediction can significantly outperform prior work on selective prediction and active learning and achieves more optimal utilization of humans in the loop.

1 INTRODUCTION

Deep Neural Networks (DNNs) have shown notable success in many applications that require complex understanding of input data (He et al., 2016a; Devlin et al., 2018), including the ones that involve high-stakes decision making (Yang, 2020). For safe deployment of DNNs, it is typically required to allow them to abstain from their predictions that are likely to be wrong, and ask humans for assistance (a task known as selective prediction) (El-Yaniv et al., 2010; Geifman & El-Yaniv, 2017). Although selective prediction can render the predictions more reliable, it does so at the cost of human interventions. For example, if a model achieves 80% accuracy on the test data, an ideal selective prediction algorithm should reject those 20% misclassified samples and send them to a human for review.

Distribution shift can significantly exacerbate the need for such human intervention. The success of DNNs often relies on the assumption that both training and test data are sampled independently and identically from the same distribution. In practice, this assumption may not hold and can degrade the performance on the

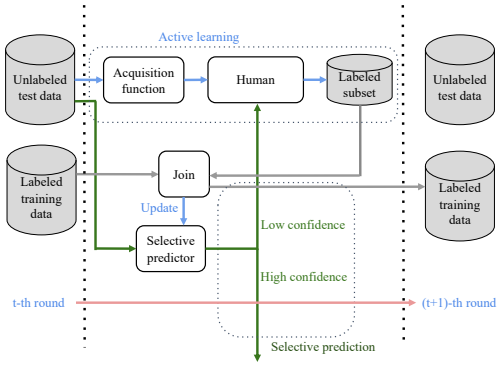

Figure 1: Illustration of our proposed problem of *active selective prediction*, where active learning is used to improve selective prediction under distribution shift. Active learning selects a small subset of data for labeling which are used to improve selective prediction on the remaining unlabeled test data. This yields more reliable predictions and more optimized use of humans in the loop.

*Work done during internship at Google.

test domain (Barbu et al., 2019; Koh et al., 2021). Existing selective prediction methods usually rely on model confidence to reject inputs (Geifman & El-Yaniv, 2017). However, model confidence can be poorly calibrated under distribution shift (Ovadia et al., 2019). Therefore, the selective classifier might accept many mis-classified test inputs, making the predictions unreliable. These render selective prediction specifically challenging under distribution shift – it either achieves an accuracy below the desired target performance, or obtains a low coverage, necessitating significant human intervention.

To improve the selective prediction performance, one idea is to rely on active learning and to have humans select a small subset of test data for labeling. The correct labels provided by humans can then be used to improve the accuracy and coverage of selective prediction on the remaining unlabeled test data, thus reducing the need for subsequent human labeling efforts. While selective prediction (Geifman & El-Yaniv, 2017; 2019) and active learning (Settles, 2009) have been studied extensively, this paper is first to propose performing active learning to improve selective prediction under distribution shift. Active domain adaptation (Su et al., 2020; Fu et al., 2021) is closest to this setting, however, it does not consider selective prediction. In selective prediction, not only does a classifier need to be learned, but a selection scoring function also needs to be constructed for rejecting misclassified inputs. Thus, going beyond conventional active learning methods that focus on selecting examples for labeling to improve the accuracy, we propose to also use those selected labeled examples to improve the selection scoring function. The optimal acquisition function (used to select examples for labeling) for this new setting is different compared to those in traditional active learning.

In this paper, we introduce a new machine learning paradigm: active selective prediction under distribution shift (see Fig. 1), which combines selective prediction and active learning to improve accuracy and coverage, and hence use human labeling in a more optimal way. Active selective prediction is highly important for most real-world deployment scenarios. To the best of our knowledge, we are the first to investigate this problem. We formulate the active selective prediction problem and propose new evaluation metrics for it. We also propose a novel and simple yet effective method, ASPEST, for solving the active selective prediction problem. The key components of ASPEST, checkpoint ensembing and self-training are proposed to address the fundamental challenges in the active selective prediction problem. On several real-world datasets, we show that ASPEST consistently outperforms other baselines proposed for active learning and selective prediction.

2 Active Selective Prediction

2.1 Problem Setup

Let $\mathcal{X}$ be the input space and $\mathcal{Y} = \{1, 2, \ldots, K\}$ the label space. The training data distribution is given as $P_{X,Y}$ and the test data distribution is $Q_{X,Y}$ (both are defined in $\mathcal{X} \times \mathcal{Y}$). There might exist distribution shifts (i.e., $Q_{X,Y}$ might be different from $P_{X,Y}$). Suppose for each input $\mathbf{x}$, an oracle (e.g., human annotator) can assign a ground-truth class label y_x to it. Given a classifier $\bar{f} : \mathcal{X} \to \mathcal{Y}$ trained on a source training dataset $\mathcal{D}^{\text{tr}} \sim P_{X,Y}$ ($\sim$ means "sampled from"), and an unlabeled target test dataset $U_X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \sim Q_X$, our goal is to employ $\bar{f}$ to yield reliable predictions on U_X in human-in-the-loop scenario. Holistically, we consider the two approaches to involve humans via the predictions they provide on the data: (i) selective prediction where uncertain predictions are deferred to humans to maintain a certain accuracy target; and (ii) active learning where a subset of U_X unlabeled samples are selected for humans to improve the model with the extra labeled data to be used at the subsequent iterations. These two approaches to involve humans have different objectives and thus, their joint optimization to best use the human labeling resources is not straightforward.

As an extension of the classifier f (initialized by $\bar{f}$), we propose to employ a selective classifier f_s including a selection scoring function $g : \mathcal{X} \to \mathbb{R}$ to yield reliable predictions on U_X. We define the predicted probability of the model f on the k-th class as $f(\mathbf{x} \mid k)$. Then, the classifier is $f(\mathbf{x}) = \arg\max_{k \in \mathcal{Y}} f(\mathbf{x} \mid k)$. g can be based on statistical operations on the outputs of f (e.g., $g(\mathbf{x}) = \max_{k \in \mathcal{Y}} f(\mathbf{x} \mid k)$). With f and g, the selective prediction model f_s is defined as:

$$f_s(\mathbf{x}; \tau) = \begin{cases} f(\mathbf{x}) & \text{if } g(\mathbf{x}) \geq \tau, \\ \perp & \text{if } g(\mathbf{x}) < \tau \end{cases}, \tag{1}$$

where τ is a threshold. If $f_s(\mathbf{x}) = \perp$, then the DNN system would defer the predictions to humans. To improve the overall accuracy to reach the target, such deferrals require manual labeling. To reduce

the labeling cost and improve the accuracy of the selective classifier, we consider labeling a small subset of U_X and adapt the selective classifier f_s on the labeled subset via active learning. The goal is to significantly improve the accuracy and coverage of f_s and thus reduce the human labeling effort.

Suppose the labeling budget for active learning is M (i.e., M examples are selected to be labeled from U_X to improve the selective prediction performance). We assume that the human in the loop can provide the correct labels. For active learning, we consider the transductive learning paradigm (Vapnik, 1998), which assumes all training and test data are observed beforehand and we can make use of the unlabeled test data for learning. Specifically, the active learning is performed on U_X to build the selective classifier f_s, with performance evaluation of f_s only on U_X. We don't consider training f_s from scratch, but adapt the source-trained classifier $\bar{f}$ to obtain f_s to maintain feasibly-low computational cost (e.g., by fine-tuning $\bar{f}$ on the M labeled data points from U_X).

Let's first consider the single-round setting. Suppose the acquisition function is $a : \mathcal{X}^m \times \mathcal{F} \times \mathcal{G} \to \mathbb{R}$, where $m \in \mathbb{N}^+$, $\mathcal{F}$ is the classifier space and $\mathcal{G}$ is the selection scoring function space. This acquisition function is the same as the one used in active learning literature (Gal et al., 2017) (refer to Appendix D for some examples of the function a). In the beginning, f is initialized by $\bar{f}$. We then select a batch B^* for labeling by solving the following objective: $B^* = \arg\max_{B \subset U_X, |B|=M} a(B, f, g)$, for which the labels are obtained to get $\tilde{B}^*$. Then, we use $\tilde{B}^*$ to update f and g (e.g., via fine-tuning the model on $\tilde{B}^*$). This can be extended to a multi-round setting. Suppose we have T rounds and the labeling budget for each round is $m = [\frac{M}{T}]$. In the beginning, f_0 is initialized by $\bar{f}$. At the t-th round, we first select a batch B_t^* for labeling by solving the following objective:

$$B_t^* = \arg \max_{B \subset U_X \setminus (\cup_{l=1}^{t-1} B_l^*), |B|=m} a(B, f_{t-1}, g_{t-1}), \tag{2}$$

for which the labels are obtained to get $\tilde{B}_t^*$. Then we use $\tilde{B}_t^*$ to update f_{t-1} and g_{t-1} to get f_t and g_t (e.g., via fine-tuning the model on $\tilde{B}_t^*$). With multiple-rounds setting, we define $B^* = \cup_{i=1}^{T} B_i^*$.

2.2 Evaluation Metrics

To quantify the efficacy of the active selective prediction methods, appropriate metrics are needed. The performance of the selective classifier f_s (defined in Eq. (1)) is evaluated by the accuracy and coverage metrics. The accuracy and coverage of f_s is defined as:

$$acc(f_s, \tau) = \frac{\mathbb{E}_{\mathbf{x} \sim U_X} \mathbb{I}[f(\mathbf{x}) = y_x \wedge g(\mathbf{x}) \geq \tau \wedge \mathbf{x} \notin B^*]}{\mathbb{E}_{\mathbf{x} \sim U_X} \mathbb{I}[g(\mathbf{x}) \geq \tau \wedge \mathbf{x} \notin B^*]} \tag{3}$$

$$cov(f_s, \tau) = \frac{\mathbb{E}_{\mathbf{x} \sim U_X} \mathbb{I}[g(\mathbf{x}) \geq \tau \wedge \mathbf{x} \notin B^*]}{\mathbb{E}_{\mathbf{x} \sim U_X} \mathbb{I}[\mathbf{x} \notin B^*]} \tag{4}$$

The following evaluation metrics are proposed to be agnostic to τ:

Maximum Accuracy at a Target Coverage t_c. The metric $acc|cov \geq t_c$ is defined as:

$$\max_{\tau} \quad acc(f_s, \tau), \quad s.t. \quad cov(f_s, \tau) \geq t_c \tag{5}$$

Maximum Coverage at a Target Accuracy t_a. The metric $cov|acc \geq t_a$ is defined as:

$$\max_{\tau} \quad cov(f_s, \tau), \quad s.t. \quad acc(f_s, \tau) \geq t_a \tag{6}$$

Area Under the Accuracy-Coverage Curve. We define the Area Under the Curve (AUC) as:

$$\text{AUC}(f_s) = \int_0^1 acc(f_s, \tau) \mathrm{d} cov(f_s, \tau) \tag{7}$$

3 Proposed Method: ASPEST

For active selective prediction, we propose utilizing active learning to improve the coverage and accuracy of the selective classifier f_s, that consists of a classifier f and a selection scoring function g.

Algorithm 1 ASPEST:Active Selective Prediction using Ensembles and Self-Training

Input: A training set $\mathcal{D}^{tr}$, a unlabeled test set U_X, the number of rounds T, the labeling budget M, the number of models N, the number of initial training steps n_s, the initial checkpoint steps c_s, a checkpoint epoch c_e, a threshold η, a sub-sampling fraction p, and a hyper-parameter λ.

Let $f_0^j = \bar{f}$ for $j = 1, \ldots, N$, and set $N_e = 0$ and $P = \mathbf{0}_{n \times K}$.

Fine-tune each f_0^j for n_s steps using obj. (9) and update P and N_e using Eq. (8) every c_s steps.

for $t = 1, \cdots, T$ **do**

Select a batch B_t from U_X for labeling using the sample selection obj. (10).

Use an oracle to assign ground-truth labels to the examples in B_t to get $\tilde{B}_t$.

Set $N_e = 0$ and $P = \mathbf{0}_{n \times K}$.

Fine-tune each f_{t-1}^j using obj. (11) to get f_t^j, while updating P and N_e using Eq (8) every c_e epochs.

Construct the pseudo-labeled set R via Eq (12) and create R_{sub} by randomly sampling up to $\lfloor p \cdot n \rfloor$ data points from R.

Train each f_t^j further using obj. (13) and update P and N_e using Eq (8) every c_e epochs.

end for

Output: $f(\mathbf{x}_i) = \arg\max_{k \in \mathcal{Y}} P_{i,k}$ and $g(\mathbf{x}_i) = \max_{k \in \mathcal{Y}} P_{i,k}$.

Different from conventional active learning, active selective prediction also aims to improve g so that it can accept those examples where f predicts correctly and reject those where f predicts incorrectly.

With distribution shift and the small amount of labeled test data, it can be challenging to train f for very high test accuracy. Therefore, g is critical to achieve high coverage and accuracy of f_s, for which we consider the confidence of f (i.e., the maximum softmax probability of f) and train f such that it has calibrated confidence. Since it has been observed that an ensemble of DNNs (known as 'deep ensembles') usually produces confidence that is more calibrated compared to a single DNN Lakshminarayanan et al. (2017), we consider f to be deep ensembles and g to be the confidence of the ensemble. Empirically, we observe that in active selective prediction, when fine-tuning the model on the small amount of selected labeled test data, the model can suffer over-fitting and over-confidence issues and ensembling the checkpoints in the training path can effectively alleviate these issues (see the analysis in Appendix F.10). Based on this observation, we propose to use checkpoint ensemble to address the challenges in active selective prediction. Since the checkpoint ensemble can have better prediction accuracy and better confidence calibration, the softmax outputs of the checkpoint ensemble can be accurate soft pseudo-labels for self-training to improve selective prediction performance. Furthermore, since the checkpoint ensemble has better uncertainty estimation, those test samples that have lowest margin of the checkpoint ensemble can be informative samples for improving selective prediction performance. Motivated by these, we propose a novel method called Active Selective Prediction using Ensembles and Self-training (ASPEST) to address the fundamental challenges in active selective prediction. Algorithm 1 presents the overall method. The details of the ASPEST algorithm are explained in Appendix C.

Since our paper is the first to study the active selective prediction problem, we begin with the simple but efficient supervised training setting. This is not a limitation of our work since the supervised training objectives can be easily extended to the semi-supervised training objectives that utilize unsupervised domain adaption techniques (see Appendix F.11).

4 EXPERIMENTS

This section presents experimental results to answer the following questions: (**Q1**) Can we use a small labeling budget for active learning to significantly improve selective prediction performance under distribution shift? (**Q2**) Does ASPEST outperform baselines across different datasets with distribution shift? (**Q3**) What is the effect of checkpoint ensembles and self-training in ASPEST?

4.1 SETUP

In this section, we briefly describe the experimental setup. More details of the setup are in Appendix E.

Dataset	DomainNet R→C (easy)		Amazon Review		Otto	
Metric	$cov\|acc \geq 80\% \uparrow$	AUC ↑	$cov\|acc \geq 80\% \uparrow$	AUC ↑	$cov\|acc \geq 80\% \uparrow$	AUC ↑
SR+Uniform	25.56±0.6	63.31±0.4	13.71±11.3	72.71±1.5	63.58±0.7	84.46±0.2
SR+Confidence	25.96±0.2	64.20±0.6	11.28±8.9	72.89±0.7	69.63±1.7	85.91±0.3
SR+Entropy	25.44±1.0	63.52±0.6	5.55±7.8	71.96±1.6	67.79±0.8	85.41±0.3
SR+Margin	26.28±1.2	64.37±0.8	14.48±10.9	73.25±1.0	68.10±0.1	85.56±0.1
SR+kCG	21.12±0.3	58.88±0.0	20.02±11.0	72.34±3.2	64.84±0.7	85.08±0.2
SR+CLUE	27.17±0.8	64.38±0.6	4.15±5.9	73.43±0.4	68.21±1.2	85.82±0.3
SR+BADGE	27.78±0.8	64.90±0.5	22.58±0.4	73.80±0.6	67.23±1.0	85.41±0.3
DE+Uniform	30.82±0.8	67.60±0.4	34.35±1.4	76.20±0.3	70.74±0.5	86.78±0.1
DE+Entropy	29.13±0.9	67.48±0.3	31.74±1.4	75.98±0.4	75.71±0.3	87.87±0.1
DE+Confidence	29.90±0.8	67.45±0.3	35.12±1.8	76.63±0.2	75.52±0.2	87.84±0.1
DE+Margin	31.82±1.3	68.85±0.4	33.42±1.3	76.18±0.2	75.49±0.8	87.89±0.2
DE+Avg-KLD	32.23±0.2	68.73±0.2	33.03±1.5	76.21±0.4	75.91±0.2	87.89±0.0
DE+CLUE	30.80±0.3	67.82±0.2	33.92±3.0	76.27±0.6	69.66±0.5	86.67±0.1
DE+BADGE	30.16±1.3	68.46±0.3	32.23±3.7	76.13±0.7	73.23±0.2	87.55±0.1
ASPEST	**37.38**±0.1	**71.61**±0.2	**38.44**±0.7	**77.69**±0.1	**77.85**±0.2	**88.28**±0.1

Table 1: Results of comparing ASPEST to the baselines on DomainNet R→C, Amazon Review and Otto. The mean and std of each metric over three random runs are reported. The labeling budget M is 500. All numbers are percentages. **Bold** numbers are superior results.

Datasets. We perform experiments on the following datasets with distribution shift: (i) MNIST→SVHN, (ii) CIFAR-10→CINIC-10, (iii) FMoW, (iv) Amazon Review, (v) DomainNet and (vi) Otto. Details on the datasets are described in Appendix E.2.

Active learning hyper-parameters. We evaluate different methods with different labeling budget M values on each dataset. By default, we set $T = 10$ for all methods (Appendix F.6 presents the effect of T). More details on the fine-tuning hyper-parameters can be found in Appendix E.4.

Baselines. We consider Softmax Response (SR) (Geifman & El-Yaniv, 2017) and Deep Ensembles (DE) (Lakshminarayanan et al., 2017) with various active learning sampling methods as the baselines. SR+Uniform means combining SR with an acquisition function based on randomly uniform sampling (similarly for others). Appendix D further describes the details of the baselines.

Hyper-parameters of ASPEST. We set $\lambda = 1$, $n_s = 1000$ and $N = 5$ (see Appendix F.7 for the effect of N), which are same as those for DE, for fair comparisons. For all datasets, we use $c_s = 200$, $p = 0.1$, $\eta = 0.9$, the number of self-training epochs to be 20 and $c_e = 5$. Note that we don't tune c_s, c_e, p and use the fixed values. We select η based on the performance on a validation dataset (i.e., DomainNet R→I) and use the same value across all other datasets.

4.2 Results

Impacts of combining selective prediction with active learning. We evaluate the accuracy of the source trained models on U_X of different datasets. The results in Appendix F.1 show that the models trained on $\mathcal{D}^{\text{tr}}$ suffer a performance drop on U_X. For example, the model trained on MNIST has a source test accuracy of 99.40%. However, its accuracy on U_X from SVHN is only 24.68%. In Table 12 (in Appendix F.3), we show that for a target accuracy of 90%, the coverage achieved by SR and DE without active learning is very low (nearly 0%). It means that almost all test examples need human intervention. This is a large cost since the test set of SVHN contains over 26K images. However, by combining selective prediction with active learning (e.g., using the proposed ASPEST), we only need to label 500 examples to achieve a target accuracy of 90% with a coverage of 87.5%. Thus, during active learning and selective prediction processes, only 12.5% examples from SVHN need to be labeled by humans to achieve the target accuracy of 90%. This indicates a significant reduction of human labeling cost. Similar results are observed for other datasets (see Appendix F.3).

Baseline comparisons. We compare ASPEST with the two existing selective classification methods: SR and DE with various active learning sampling approaches. The results in Table 1 (complete results on all datasets for all metrics and different labeling budgets are provided in Appendix F.2) show that ASPEST consistently outperforms the baselines across different image, text and tabular datasets.

Ablation studies. Compared to DE+Margin, ASPEST has two additional components: ensembling with checkpoints and self-training. We perform ablation experiments to analyze the effect of these. The results in Table 16 (in Appendix F.4) show that ASPEST achieves much better results than DE+Margin or applying those components alone.

REFERENCES

Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. *arXiv preprint arXiv:1906.03671*, 2019.

Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, 32, 2019.

William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9368–9377, 2018.

Wendy Kan Benjamin Bossan, Josef Feigl. Otto group product classification challenge, 2015.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, 2000.

Jiefeng Chen, Frederick Liu, Besim Avci, Xi Wu, Yingyu Liang, and Somesh Jha. Detecting errors and estimating accuracy on unlabeled data with self-training ensembles. *Advances in Neural Information Processing Systems*, 34:14980–14992, 2021.

Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. Functional map of the world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6172–6180, 2018.

Ching-Yao Chuang, Antonio Torralba, and Stefanie Jegelka. Estimating generalization under distribution shifts via domain-invariant representations. *arXiv preprint arXiv:2007.03511*, 2020.

Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

Sanjoy Dasgupta. Two faces of active learning. *Theoretical computer science*, 412(19):1767–1781, 2011.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Melanie Ducoffe and Frederic Precioso. Adversarial active learning for deep networks: a margin based approach. *arXiv preprint arXiv:1802.09841*, 2018.

Ran El-Yaniv et al. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(5), 2010.

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

Bo Fu, Zhangjie Cao, Jianmin Wang, and Mingsheng Long. Transferable query selection for active domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7272–7281, 2021.

Giorgio Fumera and Fabio Roli. Support vector machines with embedded reject option. In *International Workshop on Support Vector Machines*, pp. 68–82. Springer, 2002.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pp. 1183–1192. PMLR, 2017.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. *Advances in neural information processing systems*, 30, 2017.

Yonatan Geifman and Ran El-Yaniv. Selectivenet: A deep neural network with an integrated reject option. In *International conference on machine learning*, pp. 2151–2159. PMLR, 2019.

Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, 17, 2004.

Federica Granese, Marco Romanelli, Daniele Gorla, Catuscia Palamidessi, and Pablo Piantanida. Doctor: A simple method for detecting misclassification errors. *Advances in Neural Information Processing Systems*, 34: 5669–5681, 2021.

Guy Hacohen, Avihu Dekel, and Daphna Weinshall. Active learning on a budget: Opposite strategies suit high and low budgets. *arXiv preprint arXiv:2202.02794*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

Martin E Hellman. The nearest neighbor classification rule with a reject option. *IEEE Transactions on Systems Science and Cybernetics*, 6(3):179–185, 1970.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017a.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017b.

Sheng-Jun Huang, Rong Jin, and Zhi-Hua Zhou. Active learning by querying informative and representative examples. *Advances in neural information processing systems*, 23, 2010.

Amita Kamath, Robin Jia, and Percy Liang. Selective question answering under domain shift. *arXiv preprint arXiv:2006.09462*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Andreas Kirsch, Tom Rainforth, and Yarin Gal. Test distribution-aware active learning: A principled approach against distribution shift and outliers. *arXiv preprint arXiv:2106.11719*, 2021.

Sosuke Kobayashi, Shun Kiyono, Jun Suzuki, and Kentaro Inui. Diverse lottery tickets boost ensemble from a single pretrained model. *arXiv preprint arXiv:2205.11833*, 2022.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pp. 5637–5664. PMLR, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Yann LeCun. The MNIST database of handwritten digits. 1998. URL `http://yann.lecun.com/exdb/mnist/`.

Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.

Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, pp. 896, 2013.

Xiaofeng Liu, Chaehwa Yoo, Fangxu Xing, Hyejin Oh, Georges El Fakhri, Je-Won Kang, Jonghye Woo, et al. Deep unsupervised domain adaptation: A review of recent advances and perspectives. *APSIPA Transactions on Signal and Information Processing*, 11(1), 2022.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. *Advances in neural information processing systems*, 31, 2018.

Andrew McCallum, Kamal Nigam, et al. Employing em and pool-based active learning for text classification. In *ICML*, volume 98, pp. 350–358. Citeseer, 1998.

Mohammad Moghimi, Serge J Belongie, Mohammad J Saberian, Jian Yang, Nuno Vasconcelos, and Li-Jia Li. Boosted convolutional neural networks. In *BMVC*, volume 5, pp. 6, 2016.

Hussein Mozannar and David Sontag. Consistent estimators for learning to defer to an expert. In *International Conference on Machine Learning*, pp. 7076–7087. PMLR, 2020.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 188–197, 2019.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1406–1415, 2019.

Viraj Prabhu, Arjun Chandrasekaran, Kate Saenko, and Judy Hoffman. Active domain adaptation via clustering uncertainty-weighted embeddings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8505–8514, 2021.

Stephan Rabanser, Anvith Thudi, Kimia Hamidieh, Adam Dziedzic, and Nicolas Papernot. Selective classification via neural network training dynamics. *arXiv preprint arXiv:2205.13532*, 2022.

Kuniaki Saito, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Semi-supervised domain adaptation via minimax entropy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8050–8058, 2019.

Mohammadreza Salehi, Hossein Mirzaei, Dan Hendrycks, Yixuan Li, Mohammad Hossein Rohban, and Mohammad Sabokrou. A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges. *arXiv preprint arXiv:2110.14051*, 2021.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

Burr Settles. Active learning literature survey. 2009.

Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5972–5981, 2019.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608, 2020.

Jong-Chyi Su, Yi-Hsuan Tsai, Kihyuk Sohn, Buyu Liu, Subhransu Maji, and Manmohan Chandraker. Active adversarial domain adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 739–748, 2020.

Vladimir Vapnik. *Statistical learning theory*. Wiley, 1998. ISBN 978-0-471-03003-4.

Feng Wang, Guoyizhe Wei, Qiao Liu, Jinxiang Ou, Hairong Lv, et al. Boost neural networks by checkpoints. *Advances in Neural Information Processing Systems*, 34:19719–19729, 2021.

Colin Wei, Kendrick Shen, Yining Chen, and Tengyu Ma. Theoretical analysis of self-training with deep networks on unlabeled data. In *International Conference on Learning Representations*, 2020.

Wanqian Yang. *Making Decisions Under High Stakes: Trustworthy and Expressive Bayesian Deep Learning*. PhD thesis, 2020.

Huaxiu Yao, Caroline Choi, Bochuan Cao, Yoonho Lee, Pang Wei Koh, and Chelsea Finn. Wild-time: A benchmark of in-the-wild distribution shift over time. *arXiv preprint arXiv:2211.14238*, 2022.

David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pp. 189–196, 1995.

Michelle Yuan, Hsuan-Tien Lin, and Jordan Boyd-Graber. Cold-start active learning through self-supervised language modeling. *arXiv preprint arXiv:2010.09535*, 2020.

Eric Zhao, Anqi Liu, Animashree Anandkumar, and Yisong Yue. Active learning under label shift. In *International Conference on Artificial Intelligence and Statistics*, pp. 3412–3420. PMLR, 2021.

Xiatian Zhu, Shaogang Gong, et al. Knowledge distillation by on-the-fly native ensemble. *Advances in neural information processing systems*, 31, 2018.

Supplementary Material

In Section A, we discuss some potential negative societal impacts of our work. In Section B, we discuss related work. In Section C, we describe the proposed algorithm ASPEST for the active selective prediction problem and then analyze its computational complexity. In Section D, we describe the baselines in detail. In Section E, we provide the details of the experimental setup. In Section F, we give some additional experimental results.

A Potential Negative Societal Impacts

The proposed framework yields more reliable predictions with more optimized utilization of humans in the loop. One potential risk of such a system is that if the humans in the loop yield inaccurate or biased labels, our framework might cause them being absorbed by the predictor model and the selection prediction mechanism, and eventually the outcomes of the system might be inaccurate and biased. We leave the methods for inaccurate label or bias detection to future work.

B Related Work

Selective prediction. Selective prediction (also known as prediction with rejection/deferral options) constitutes a common deployment scenario for DNNs, especially in high-stakes decision making scenarios. In selective prediction, models abstain from yielding outputs if their confidence on the likelihood of correctness is not sufficiently high. Such abstinence usually incurs deferrals to humans and results in additional cost (Mozannar & Sontag, 2020). Increasing the coverage – the ratio of the samples for which the DNN outputs can be reliable – is the fundamental goal (El-Yaniv et al., 2010; Fumera & Roli, 2002; Hellman, 1970; Geifman & El-Yaniv, 2019). Geifman & El-Yaniv considers selective prediction for DNNs with the 'Softmax Response' method, which applies a carefully selected threshold on the maximal response of the softmax layer to construct the selective classifier. Lakshminarayanan et al. shows that using deep ensembles can improve predictive uncertainty estimates and thus improve selective prediction. Rabanser et al. proposes a novel method, NNTD, for selective prediction that utilizes DNN training dynamics by using checkpoints during training. Our proposed method ASPEST also uses checkpoints to construct ensembles for selective prediction. In contrast to NNTD and other aforementioned methods, we combine selective prediction with active learning to improve its data efficiency while considering a holistic perspective of having humans in the loop. This new active selective prediction setup warrants new methods for selective prediction along with active learning.

Active learning. To utilize the human labeling budget more effectively while training DNNs, active learning employs acquisition functions to select unlabeled examples for labeling, and uses these labeled examples to train models (Settles, 2009; Dasgupta, 2011). Commonly-used active learning methods employ acquisition functions by considering uncertainty (Gal et al., 2017; Ducoffe & Precioso, 2018; Beluch et al., 2018) or diversity (Sener & Savarese, 2017; Sinha et al., 2019), or their combination (Ash et al., 2019; Huang et al., 2010). One core challenge for active learning is the "cold start" problem: often the improved obtained from active learning is less significant when the amount of labeled data is significantly smaller (Yuan et al., 2020; Hacohen et al., 2022). Moreover, active learning can be particularly challenging under distribution shift (Kirsch et al., 2021; Zhao et al., 2021). Recently, active domain adaptation has been studied, where domain adaptation is combined with active learning (Su et al., 2020; Fu et al., 2021; Prabhu et al., 2021). Different from traditional active learning, active domain adaptation typically adapts a model pre-trained on the labeled source domain to the unlabeled target domain. In our work, we also try to adapt a source trained model to the unlabeled target test set using active learning, while focusing on building a selective classification model and reducing the human labeling effort.

Distribution shift. Distribution shift, where the training distribution differs from the test distribution, often occurs in practice and can substantially degrade the accuracy of the deployed DNNs (Koh et al., 2021; Yao et al., 2022; Barbu et al., 2019). Distribution shift can also substantially reduce the quality of uncertainty estimation (Ovadia et al., 2019), which is often used for rejecting examples in selective prediction and selecting samples for labeling in active learning. Several techniques try to tackle the challenge caused by distribution shift, including accuracy estimation (Chen et al., 2021; Chuang

et al., 2020), error detection (Hendrycks & Gimpel, 2016; Granese et al., 2021), out-of-distribution detection (Salehi et al., 2021), domain adaptation (Ganin et al., 2016; Saito et al., 2019), selective prediction (Kamath et al., 2020) and active learning (Kirsch et al., 2021). In our work, we combine selective prediction with active learning to address the issue of distribution shift.

Deep ensembles. Ensembles of DNNs (or deep ensembles) have been successfully used to boost predictive performance (Moghimi et al., 2016; Zhu et al., 2018). Deep ensembles can also be used to improve the predictive uncertainty estimation (Lakshminarayanan et al., 2017; Fort et al., 2019). Lakshminarayanan et al. shows that random initialization of the NN parameters along with random shuffling of the data points are sufficient for deep ensembles to perform well in practice. However, training multiple DNNs from random initialization can be very expensive. To obtain deep ensembles more efficiently, recent papers explore using checkpoints during training to construct the ensemble (Wang et al., 2021; Huang et al., 2017a), or fine-tuning a single pre-trained model to create the ensemble (Kobayashi et al., 2022). In our work, we use the checkpoints during fine-tuning a source-trained model via active learning as the ensemble and further boost the ensemble's performance via self-training. We also use the ensemble's uncertainty measured by a margin to select samples for labeling in active learning.

Self-training. Self-training is a common algorithmic paradigm for leveraging unlabeled data with DNNs. Self-training methods train a model to fit pseudo-labels (i.e., predictions on unlabeled data made by a previously-learned model) to boost the model's performance (Yarowsky, 1995; Grandvalet & Bengio, 2004; Lee et al., 2013; Wei et al., 2020; Sohn et al., 2020). In this work, we use self-training to improve the selective prediction performance of deep ensembles. Instead of using predicted labels as pseudo-labels as a common practice in prior works, we use the average softmax outputs of the checkpoints during training as the pseudo-labels and self-train the models in the ensemble on them with the KL-Divergence loss to improve the selective prediction performance.

C ASPEST ALGORITHM

In this section, we describe the proposed algorithm ASPEST for the active selective prediction problem and then analyze its computational complexity.

C.1 ALGORITHM DESIGN

For active selective prediction, we propose utilizing active learning to improve the coverage and accuracy of the selective classifier f_s, that consists of a classifier f and a selection scoring function g. Different from conventional active learning, which only aims to improve the accuracy of the classifier f, active selective prediction also aims to improve g so that it can accept those examples where f predicts correctly and reject those where f predicts incorrectly.

With distribution shift and the small amount of labeled test data, it can be challenging to train f for very high test accuracy. Therefore, g is critical to achieve high coverage and accuracy of f_s, for which we consider the confidence of f (i.e., the maximum softmax probability of f) and train f such that it has calibrated confidence. Since it has been observed that an ensemble of DNNs (known as 'deep ensembles') usually produces confidence that is more calibrated compared to a single DNN (Lakshminarayanan et al., 2017), we consider f to be deep ensembles and g to be the confidence of the ensemble. Empirically, we observe that in active selective prediction, when fine-tuning the model on the small amount of selected labeled test data, the model can suffer over-fitting and over-confidence issues and ensembling the checkpoints in the training path can effectively alleviate these issues (see the empirical analysis in Appendix F.10). Based on this observation, we propose to use checkpoint ensemble to address the challenges in active selective prediction. Since the checkpoint ensemble can have better prediction accuracy and better confidence calibration, the softmax outputs of the checkpoint ensemble can be accurate soft pseudo-labels for self-training to improve selective prediction performance. Furthermore, since the checkpoint ensemble has better uncertainty estimation, those test samples that have lowest margin of the checkpoint ensemble can be informative samples for improving selective prediction performance. Motivated by these, we propose a novel method called Active Selective Prediction using Ensembles and Self-training (ASPEST) to address the fundamental challenges in active selective prediction. Algorithm 1 presents the overall method and the details are explained next.

Constructing checkpoint ensemble. We first describe how the intermediate model checkpoints (or weights) during training are used to construct the checkpoint ensemble. Since we have all the test inputs, we don't need to save the checkpoints during training, but just record their outputs on the test set U_X. Specifically, we use a $n \times K$ matrix P (recall that $n = |U_X|$ and K is the number of classes) to record the average of the softmax outputs of the checkpoint ensemble and use N_e to record the number of checkpoints in the current checkpoint ensemble. Given a checkpoint model f, we update P and N_e via:

$$\begin{aligned} P_{i,k} &\leftarrow \frac{1}{N_e+1}(P_{i,k} \cdot N_e + f(\mathbf{x}_i \mid k)), \text{ for } i = 1, \ldots, n \text{ and } k = 1, \ldots, K \\ N_e &\leftarrow N_e + 1. \end{aligned} \tag{8}$$

Ensemble training. We continue fine-tuning N models independently via Stochastic Gradient Descent (SGD) with different random seeds (e.g., the randomness can come from different random orders of training batches). At the beginning, we set each model $f_0^j = \bar{f}$ $(j = 1, \ldots, N)$, and set $N_e = 0$ and $P = \mathbf{0}_{n \times K}$. Here, we initialize each model f_0^j with $\bar{f}$ instead of random initialization, to minimize the computational cost. We fine-tune each model f_0^j on $\mathcal{D}^{\text{tr}}$ for n_s steps via SGD using the following training objective:

$$\min_{\theta^j} \quad \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta^j), \tag{9}$$

where ℓ_{CE} is the cross-entropy loss and θ^j is the model parameters of f_0^j. For every c_s steps when training each f_0^j, we update P and N_e using Eq (8) with the checkpoint model f_0^j.

After constructing the initial checkpoint ensemble, we perform a T-round active learning process. In each round of active learning, we first select samples for labeling based on the margin of the checkpoint ensemble, then fine-tune the models on the selected labeled test data, and finally perform self-training. We describe the procedure below:

Sample selection. In the t-th round, we select a batch B_t with a size of $m = [\frac{M}{T}]$ from U_X for labeling via:

$$B_t = \arg \max_{B \subset U_X \setminus (\cup_{l=0}^{t-1} B_l), |B|=m} - \sum_{x_i \in B} S(\mathbf{x}_i) \tag{10}$$

where $B_0 = \emptyset$ and $S(\mathbf{x}_i)$ is defined as:

$$S(\mathbf{x}_i) = P_{i,\hat{y}} - \max_{k \in \mathcal{Y} \setminus \{\hat{y}\}} P_{i,k}, \quad s.t. \quad \hat{y} = \arg\max_{k \in \mathcal{Y}} P_{i,k}.$$

We use an oracle to assign ground-truth labels to the examples in B_t to get $\tilde{B}_t$. Here, we select the test samples for labeling based on the margin of the checkpoint ensemble. The test samples with lower ensemble margins should be closer to the decision boundary of the ensemble and they are the data points where the ensemble is uncertain about its predictions. Training on those data points can either make the predictions of the ensemble more accurate or make the ensemble have higher confidence on its correct predictions. Thus, selecting those data points with the lowest ensemble margin for labeling and training on them can improve the selective prediction performance of the ensemble.

Fine-tuning. After the sample selection, we reset N_e and P via: $N_e = 0$ and $P = \mathbf{0}_{n \times K}$. We reset N_e and P because we want to remove those checkpoints in the previous rounds with a worse performance from the checkpoint ensemble (experiments in Appendix F.9 show that after each round of active learning, the accuracy of the ensemble will be significantly improved). We then fine-tune each model f_{t-1}^j $(j = 1, \ldots, N)$ independently via SGD with different randomness on the selected labeled test data to get f_t^j using the following training objective:

$$\min_{\theta^j} \quad \mathbb{E}_{(\mathbf{x},y)\in\cup_{l=1}^{t}\tilde{B}_l} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) + \lambda \cdot \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) \tag{11}$$

where θ^j is the model parameters of f_{t-1}^j and λ is a hyper-parameter. Note that here we use joint training on $\mathcal{D}^{\text{tr}}$ and $\cup_{l=1}^{t}\tilde{B}_l$ to avoid over-fitting to the small set of labeled test data and prevent the models from forgetting the source training knowledge (see the results in Appendix F.5 for the effect

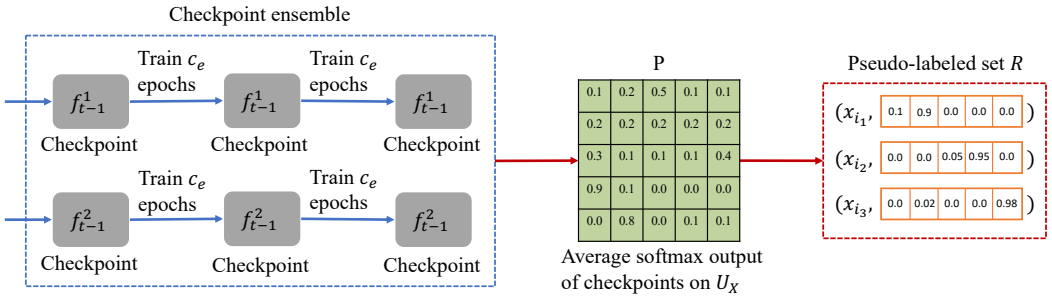

Figure 2: Illustration of the checkpoint ensemble and pseudo-labeled set construction in the proposed method ASPEST.

of using joint training and the effect of λ). For every c_e epoch when fine-tuning each model f^j_{t-1}, we update P and N_e using Eq. (8) with the checkpoint model f^j_{t-1}.

Self-training. After fine-tuning the models on the selected labeled test data and with the checkpoint ensemble, we construct a pseudo-labeled set R (see Fig. 2) via:

$$R = \{(\mathbf{x}_i, P_{i,:}) \mid \mathbf{x}_i \in U_X \wedge (\eta \leq \max_{k \in \mathcal{Y}} P_{i,k} < 1)\}, \tag{12}$$

where $\max_{k\in\mathcal{Y}} P_{i,k}$ is the confidence of the checkpoint ensemble on $\mathbf{x}_i$ and η is a threshold (refer to Section 4.2 for the effect of η). We do not add those test data points with confidence equal to 1 into the pseudo-labeled set because training on those data points cannot change the models much and may even hurt the performance (refer to Appendix F.8 for the justification of such a design). We then perform self-training on the pseudo-labeled set R. For computational efficiency, we only apply self-training on a subset of R. We construct the subset R_{sub} by randomly sampling up to $\lfloor p \cdot n \rfloor$ data points from R, where $p \in [0, 1]$. We train each model f^j_t $(j = 1, \ldots, N)$ further on the pseudo-labeled subset R_{sub} via SGD using the following training objective:

$$\min_{\theta^j} \quad \mathbb{E}_{(\mathbf{x},\mathbf{y})\in R_{\text{sub}}} \quad \ell_{KL}(\mathbf{x}, \mathbf{y}; \theta^j) + \lambda \cdot \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) \tag{13}$$

where ℓ_{KL} is the KL-Divergence loss, which is defined as:

$$\ell_{KL}(\mathbf{x}, \mathbf{y}; \theta) = \sum_{k=1}^{K} \mathbf{y}_k \cdot \log(\frac{\mathbf{y}_k}{f(\mathbf{x} \mid k; \theta)}) \tag{14}$$

Note that for self-training, we typically use predicted labels as pseudo-labels and the cross-entropy loss. We don't follow this because the predicted labels might be wrong and training the models on those misclassified pseudo-labeled data points using the cross entropy loss will make the models have very high confidence on their wrong predictions, which will hurt the selective prediction performance.

For every c_e epoch of self-training each model f^j_t, we will update P and N_e using Eq (8) with the checkpoint model f^j_t. We add checkpoints during self-training into checkpoint ensemble to improve the sample selection in the next round of active learning. In our experiments, we demonstrate that such self-training can significantly improve the selective prediction performance (see Section 4.2).

Final selective classifier. After T rounds active learning, we output the classifier $f(\mathbf{x}_i) = \arg\max_{k\in\mathcal{Y}} P_{i,k}$ and the selection scoring function $g(\mathbf{x}_i) = \max_{k\in\mathcal{Y}} P_{i,k}$.

C.2 Computational Complexity

Let the complexity for one step of updating P and N_e be t_u (mainly one forward pass of DNN); for one DNN gradient update step is t_g (mainly one forward and backward pass of DNN); and for sample selection is t_s (mainly sorting test examples). Then, the total complexity of ASPEST would be $O\Big(N \cdot \frac{n_s}{c_s} \cdot t_u + N \cdot n_s \cdot t_g + T \cdot [t_s + N \cdot (e_f + e_s \cdot p) \cdot \frac{n}{b} \cdot t_g + N \cdot \frac{e_s+e_f}{c_e} \cdot t_u]\Big)$, where e_f is the number of fine-tuning epochs and e_s is the number of self-training epochs and b is the batch

size. Although the training objectives include training on $\mathcal{D}^{\text{tr}}$, the complexity doesn't depend on the size of $\mathcal{D}^{\text{tr}}$ since we measure e_f over $\cup_{l=1}^{t}\tilde{B}_l$ in training objective (11) and measure e_s over R_{sub} in training objective (13). In practice, we usually have $t_s \ll t_g$ and $t_u \ll t_g$. Also, we set $e_s \cdot p \ll e_f$, $n_s \ll \frac{n}{b} \cdot T \cdot e_f$ and $\frac{e_s+e_f}{c_e} \ll (e_f + e_s \cdot p) \cdot \frac{n}{b}$. So the complexity of ASPEST is $O\Big(N \cdot T \cdot \frac{n}{b} \cdot e_f \cdot t_g\Big)$. Suppose the size of $\mathcal{D}^{\text{tr}}$ is n_{tr} and the number of source training epochs is e_p. Then, the complexity for source training is $O(\frac{n_{tr}}{b} \cdot e_p \cdot t_g)$. In practice, we usually have $N \cdot T \cdot n \cdot e_f \ll n_{tr} \cdot e_p$. Overall, the complexity of ASPEST would be much smaller than that of source training.

D BASELINES

We consider two selective classification baselines Softmax Response (SR) (Geifman & El-Yaniv, 2017) and Deep Ensembles (DE) (Lakshminarayanan et al., 2017) and combine them with active learning techniques. We describe them in detail below.

D.1 SOFTMAX RESPONSE

Suppose the neural network classifier is f where the last layer is a softmax. Let $f(\mathbf{x} \mid k)$ be the soft response output for the k-th class. Then the classifier is defined as $f(\mathbf{x}) = \arg\max_{k\in\mathcal{Y}} f(\mathbf{x} \mid k)$ and the selection scoring function is defined as $g(\mathbf{x}) = \max_{k\in\mathcal{Y}} f(\mathbf{x} \mid k)$, which is also known as the Maximum Softmax Probability (MSP) of the neural network. Recall that with f and g, the selective classifier is defined in Eq (1). We use active learning to fine-tune the model f to improve the selective prediction performance of Softmax Response on the unlabeled test dataset U_X. The complete algorithm is presented in Algorithm 2. In our experiments, we always set $\lambda = 1$. We use the joint training objective (23) to avoid over-fitting to the small labeled test set $\cup_{l=1}^{t}\tilde{B}_l$ and prevent the model from forgetting the source training knowledge. The algorithm can be combined with different kinds of acquisition functions. We describe the acquisition functions considered for Softmax Response below.

Uniform. In the t-th round of active learning, we select $[\frac{M}{T}]$ data points as the batch B_t from $U_X \setminus \cup_{l=0}^{t-1}B_l$ via uniform random sampling. The corresponding acquisition function is: $a(B, f_{t-1}, g_{t-1}) = 1$. When solving the objective (22), the tie is broken randomly.

Confidence. We define the confidence score of f on the input $\mathbf{x}$ as

$$S_{\text{conf}}(\mathbf{x}; f) = \max_{k\in\mathcal{Y}} \quad f(\mathbf{x} \mid k) \tag{15}$$

Then the acquisition function in the t-th round of active learning is defined as:

$$a(B, f_{t-1}, g_{t-1}) = -\sum_{\mathbf{x}\in B} S_{\text{conf}}(\mathbf{x}; f_{t-1}) \tag{16}$$

That is we select those test examples with the lowest confidence scores for labeling.

Entropy. We define the entropy score of f on the input $\mathbf{x}$ as

$$S_{\text{entropy}}(\mathbf{x}; f) = \sum_{k\in\mathcal{Y}} \quad -f(\mathbf{x} \mid k) \cdot \log f(\mathbf{x} \mid k) \tag{17}$$

Then the acquisition function in the t-th round of active learning is defined as:

$$a(B, f_{t-1}, g_{t-1}) = \sum_{\mathbf{x}\in B} S_{\text{entropy}}(\mathbf{x}; f_{t-1}) \tag{18}$$

That is we select those test examples with the highest entropy scores for labeling.

Margin. We define the margin score of f on the input $\mathbf{x}$ as

$$S_{\text{margin}}(\mathbf{x}; f) = f(\mathbf{x} \mid \hat{y}) - \max_{k\in\mathcal{Y}\setminus\{\hat{y}\}} \quad f(\mathbf{x} \mid k) \tag{19}$$

$$s.t. \quad \hat{y} = \arg\max_{k\in\mathcal{Y}} \quad f(\mathbf{x} \mid k) \tag{20}$$

Then the acquisition function in the t-th round of active learning is defined as:

$$a(B, f_{t-1}, g_{t-1}) = -\sum_{\mathbf{x} \in B} S_{\text{margin}}(\mathbf{x}; f_{t-1}) \tag{21}$$

That is we select those test examples with lowest margin scores for labeling.

kCG. We use the k-Center-Greedy algorithm proposed in (Sener & Savarese, 2017) to select test examples for labeling in each round.

CLUE. We use the Clustering Uncertainty-weighted Embeddings (CLUE) proposed in (Prabhu et al., 2021) to select test examples for labeling in each round. Following (Prabhu et al., 2021), we set the hyper-parameter $T = 0.1$ on DomainNet and set $T = 1.0$ on other datasets.

BADGE. We use the Diverse Gradient Embeddings (BADGE) proposed in (Ash et al., 2019) to select test examples for labeling in each round.

Algorithm 2 Softmax Response with Active Learning

Input: A training dataset $\mathcal{D}^{\text{tr}}$, an unlabeled test dataset U_X, the number of rounds T, the labeling budget M, a source-trained model $\bar{f}$, an acquisition function a and a hyper-parameter λ.
Let $f_0 = \bar{f}$.
Let $B_0 = \emptyset$.
Let $g_t(\mathbf{x}) = \max_{k \in \mathcal{Y}} f_t(\mathbf{x} \mid k)$.
for $t = 1, \cdots, T$ **do**
Select a batch B_t with a size of $m = [\frac{M}{T}]$ from U_X for labeling via:

$$B_t = \arg \max_{B \subset U_X \setminus (\cup_{l=0}^{t-1} B_l), |B| = m} a(B, f_{t-1}, g_{t-1}) \tag{22}$$

Use an oracle to assign ground-truth labels to the examples in B_t to get $\tilde{B}_t$.
Fine-tune the model f_{t-1} using the following training objective:

$$\min_{\theta} \quad \mathbb{E}_{(\mathbf{x},y) \in \cup_{l=1}^{t} \tilde{B}_l} \quad \ell_{CE}(\mathbf{x}, y; \theta) + \lambda \cdot \mathbb{E}_{(\mathbf{x},y) \in \mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta) \tag{23}$$

where θ is the model parameters of f_{t-1} and ℓ_{CE} is the cross-entropy loss function.
Let $f_t = f_{t-1}$.
end for
Output: The classifier $f = f_T$ and the selection scoring function $g = \max_{k \in \mathcal{Y}} f(\mathbf{x} \mid k)$.

D.2 Deep Ensembles

It has been shown that deep ensembles can significantly improve the selective prediction performance (Lakshminarayanan et al., 2017), not only because deep ensembles are more accurate than a single model, but also because deep ensembles yield more calibrated confidence.

Suppose the ensemble model f contains N models $f^1, \ldots, f^N$. Let $f^j(\mathbf{x} \mid k)$ denote the predicted probability of the model f^j on the k-th class. We define the predicted probability of the ensemble model f on the k-th class as:

$$f(\mathbf{x} \mid k) = \frac{1}{N} \sum_{j=1}^{N} f^j(\mathbf{x} \mid k). \tag{24}$$

The classifier is defined as $f(\mathbf{x}) = \arg\max_{k \in \mathcal{Y}} f(\mathbf{x} \mid k)$ and the selection scoring function is defined as $g(\mathbf{x}) = \max_{k \in \mathcal{Y}} f(\mathbf{x} \mid k)$. We use active learning to fine-tune each model f^j in the ensemble to improve the selective prediction performance of the ensemble on the unlabeled test dataset U_X. Each model f^j is first initialized by the source-trained model $\bar{f}$, and then fine-tuned independently via Stochastic Gradient Decent (SGD) with different sources of randomness (e.g., different random order of the training batches) on the training dataset $\mathcal{D}^{\text{tr}}$ and the selected labeled test data. Note that this way to construct the ensembles is different from the standard Deep Ensembles method, which trains the models from different random initialization. We use this way to construct the ensemble due to the constraint in our problem setting, which requires us to fine-tune a given source-trained

model $\bar{f}$. Training the models from different random initialization might lead to an ensemble with better performance, but it is much more expensive, especially when the training dataset and the model are large (e.g., training foundation models). Thus, the constraint in our problem setting is feasible in practice. The complete algorithm is presented in Algorithm 3. In our experiments, we always set $\lambda = 1$, $N = 5$, and $n_s = 1000$. We also use joint training here and the reasons are the same as those for the Softmax Response baseline. The algorithm can be combined with different kinds of acquisition functions. We describe the acquisition functions considered below.

Uniform. In the t-th round of active learning, we select $[\frac{M}{T}]$ data points as the batch B_t from $U_X \setminus \cup_{l=0}^{t-1} B_l$ via uniform random sampling. The corresponding acquisition function is: $a(B, f_{t-1}, g_{t-1}) = 1$. When solving the objective (31), the tie is broken randomly.

Confidence. The confidence scoring function S_{conf} for the ensemble model f is the same as that in Eq. (15) ($f(\mathbf{x} \mid k)$ for the ensemble model f is defined in Eq. (24)). The acquisition function in the t-th round of active learning is defined as:

$$a(B, f_{t-1}, g_{t-1}) = -\sum_{\mathbf{x} \in B} S_{\text{conf}}(\mathbf{x}; f_{t-1}) \tag{25}$$

That is we select those test examples with the lowest confidence scores for labeling.

Entropy. The entropy scoring function S_{entropy} for the ensemble model f is the same as that in Eq. (17). The acquisition function in the t-th round of active learning is defined as:

$$a(B, f_{t-1}, g_{t-1}) = \sum_{\mathbf{x} \in B} S_{\text{entropy}}(\mathbf{x}; f_{t-1}), \tag{26}$$

That is we select those test examples with the highest entropy scores for labeling.

Margin. The margin scoring function S_{margin} for the ensemble model f is the same as that in Eq. (19). The acquisition function in the t-th round of active learning is defined as:

$$a(B, f_{t-1}, g_{t-1}) = -\sum_{\mathbf{x} \in B} S_{\text{margin}}(\mathbf{x}; f_{t-1}) \tag{27}$$

That is we select those test examples with the lowest margin scores for labeling.

Avg-KLD. The Average Kullback-Leibler Divergence (Avg-KLD) is proposed in (McCallum et al., 1998) as a disagreement measure for the model ensembles, which can be used for sample selection in active learning. The Avg-KLD score of the ensemble model f on the input $\mathbf{x}$ is defined as:

$$S_{\text{kl}}(\mathbf{x}; f) = \frac{1}{N} \sum_{j=1}^{N} \sum_{k \in \mathcal{Y}} f^j(\mathbf{x} \mid k) \cdot \log \frac{f^j(\mathbf{x} \mid k)}{f(\mathbf{x} \mid k)}. \tag{28}$$

Then the acquisition function in the t-th round of active learning is defined as:

$$a(B, f_{t-1}, g_{t-1}) = \sum_{\mathbf{x} \in B} S_{\text{kl}}(\mathbf{x}; f_{t-1}), \tag{29}$$

That is we select those test examples with the highest Avg-KLD scores for labeling.

CLUE. CLUE (Prabhu et al., 2021) is proposed for a single model. Here, we adapt CLUE for the ensemble model, which requires a redefinition of the entropy function $\mathcal{H}(Y \mid \mathbf{x})$ and the embedding function $\phi(\mathbf{x})$ used in the CLUE algorithm. We define the entropy function as Eq. (17) with the ensemble model f. Suppose ϕ^j is the embedding function for the model f^j in the ensemble. Then, the embedding of the ensemble model f on the input $\mathbf{x}$ is $[\phi^1(\mathbf{x}), \ldots, \phi^N(\mathbf{x})]$, which is the concatenation of the embeddings of the models $f^1, \ldots, f^N$ on $\mathbf{x}$. Following (Prabhu et al., 2021), we set the hyper-parameter $T = 0.1$ on DomainNet and set $T = 1.0$ on other datasets.

BADGE. BADGE (Ash et al., 2019) is proposed for a single model. Here, we adapt BADGE for the ensemble model, which requires a redefinition of the gradient embedding g_x in the BADGE algorithm. Towards this end, we propose the gradient embedding g_x of the ensemble model f as the concatenation of the gradient embeddings of the models $f^1, \ldots, f^N$.

Algorithm 3 Deep Ensembles with Active Learning

Input: A training dataset $\mathcal{D}^{\text{tr}}$, An unlabeled test dataset U_X, the number of rounds T, the total labeling budget M, a source-trained model $\bar{f}$, an acquisition function $a(B, f, g)$, the number of models in the ensemble N, the number of initial training steps n_s, and a hyper-parameter λ.

Let $f_0^j = \bar{f}$ for $j = 1, \ldots, N$.

Fine-tune each model f_0^j in the ensemble via SGD for n_s training steps independently using the following training objective with different randomness:

$$\min_{\theta^j} \quad \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) \tag{30}$$

where θ^j is the model parameters of f_0^j and ℓ_{CE} is the cross-entropy loss function.

Let $B_0 = \emptyset$.

Let $g_t(\mathbf{x}) = \max_{k\in\mathcal{Y}} f_t(\mathbf{x} \mid k)$.

for $t = 1, \cdots, T$ **do**

Select a batch B_t with a size of $m = [\frac{M}{T}]$ from U_X for labeling via:

$$B_t = \arg \max_{B\subset U_X \setminus (\cup_{l=0}^{t-1} B_l), |B|=m} a(B, f_{t-1}, g_{t-1}) \tag{31}$$

Use an oracle to assign ground-truth labels to the examples in B_t to get $\tilde{B}_t$.

Fine-tune each model f_{t-1}^j in the ensemble via SGD independently using the following training objective with different randomness:

$$\min_{\theta^j} \quad \mathbb{E}_{(\mathbf{x},y)\in\cup_{l=1}^{t}\tilde{B}_l} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) + \lambda \cdot \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) \tag{32}$$

where θ^j is the model parameters of f_{t-1}^j.

Let $f_t^j = f_{t-1}^j$.

end for

Output: The classifier $f = f_T$ and the selection scoring function $g = \max_{k\in\mathcal{Y}} f(\mathbf{x} \mid k)$.

E Details of Experimental Setup

E.1 Computing Infrastructure and Runtime

We run all experiments with TensorFlow 2.0 on NVIDIA A100 GPUs in the Debian GNU/Linux 10 system. We report the total runtime of the proposed method ASPEST on each dataset in Table 2. Note that in our implementation, we train models in the ensemble sequentially. However, it is possible to train models in the ensemble in parallel, which can significantly reduce the runtime. With the optimal implementation, the inference latency of the ensemble can be as low as the inference latency of a single model.

Dataset	Total Runtime
MNIST→SVHN	24 min
CIFAR-10→CINIC-10	1 hour
FMoW	2 hour 48 min
Amazon Review	1 hour 34 min
DomainNet (R→C)	2 hours 10 min
DomainNet (R→P)	1 hour 45 min
DomainNet (R→S)	1 hour 51 min
DomainNet (R→I)	1 hour 38 min
Otto	18 min

Table 2: The runtime of ASPEST when the labeling budget $M = 500$. We use the default hyper-parameters for ASPEST described in Section 4.1.

E.2 DATASETS

We describe the datasets used below. For all image datasets, we normalize the range of pixel values to [0,1].

MNIST→SVHN. The source training dataset $\mathcal{D}^{\text{tr}}$ is MNIST (LeCun, 1998) while the target test dataset U_X is SVHN (Netzer et al., 2011). MNIST consists 28×28 grayscale images of handwritten digits, containing in total 5,500 training images and 1,000 test images. We resize each image to be 32×32 resolution and change them to be colored. We use the training set of MNIST as $\mathcal{D}^{\text{tr}}$ and the test set of MNIST as the source validation dataset. SVHN consists 32×32 colored images of digits obtained from house numbers in Google Street View images. The training set has 73,257 images and the test set has 26,032 images. We use the test set of SVHN as U_X.

CIFAR-10→CINIC-10. The source training dataset $\mathcal{D}^{\text{tr}}$ is CIFAR-10 (Krizhevsky et al., 2009) while the target test dataset U_X is CINIC-10 (Darlow et al., 2018). CIFAR-10 consists 32×32 colored images with ten classes (dogs, frogs, ships, trucks, etc.), each consisting of 5,000 training images and 1,000 test images. We use the training set of CIFAR-10 as $\mathcal{D}^{\text{tr}}$ and the test set of CIFAR-10 as the source validation dataset. During training, we apply random horizontal flipping and random cropping with padding data augmentations to the training images. CINIC-10 is an extension of CIFAR-10 via the addition of downsampled ImageNet images. CINIC-10 has a total of 270,000 images equally split into training, validation, and test. In each subset (90,000 images) there are ten classes (identical to CIFAR-10 classes). There are 9,000 images per class per subset. We use a subset of the CINIC-10 test set containing 30,000 images as U_X.

FMoW. We use the FMoW-WILDS dataset from (Koh et al., 2021). FMoW-wilds is based on the Functional Map of the World dataset (Christie et al., 2018), which collected and categorized high-resolution satellite images from over 200 countries based on the functional purpose of the buildings or land in the image, over the years 2002–2018. The task is multi-class classification, where the input $\mathbf{x}$ is an RGB satellite image, the label y is one of 62 building or land use categories, and the domain d represents both the year the image was taken as well as its geographical region (Africa, the Americas, Oceania, Asia, or Europe). The training set contains 76,863 images from the years 2002-2013. The In-Distribution (ID) validation set contains 11,483 images from the years 2002-2013. The OOD test set contains 22,108 images from the years 2016-2018. We resize each image to be 96×96 resolution to save computational cost. We use the training set as $\mathcal{D}^{\text{tr}}$ and the ID validation set as the source validation dataset. During training, we apply random horizontal flipping and random cropping with padding data augmentations to the training images. We use the OOD test set as U_X.

Amazon Review. We use the Amazon Review WILDS dataset from (Koh et al., 2021). The dataset comprises 539,502 customer reviews on Amazon taken from the Amazon Reviews dataset (Ni et al., 2019). The task is multi-class sentiment classification, where the input $\mathbf{x}$ is the text of a review, the label y is a corresponding star rating from 1 to 5, and the domain d is the identifier of the reviewer who wrote the review. The training set contains 245,502 reviews from 1,252 reviewers. The In-Distribution (ID) validation set contains 46,950 reviews from 626 of the 1,252 reviewers in the training set. The Out-Of-Distribution (OOD) test set contains 100,050 reviews from another set of 1,334 reviewers, distinct from those of the training set. We use the training set as $\mathcal{D}^{\text{tr}}$ and the ID validation set as the source validation dataset. We use a subset of the OOD test set containing 22,500 reviews from 300 reviewers as U_X.

DomainNet. DomainNet (Peng et al., 2019) is a dataset of common objects in six different domains. All domains include 345 categories (classes) of objects such as Bracelet, plane, bird, and cello. We use five domains from DomainNet including: (1) Real: photos and real world images. The training set from the Real domain has 120,906 images while the test set has 52,041 images; (2) Clipart: a collection of clipart images. The training set from the Clipart domain has 33,525 images while the test set has 14,604 images; (3) Sketch: sketches of specific objects. The training set from the Sketch has 48,212 images while the test set has 20,916 images; (4) Painting: artistic depictions of objects in the form of paintings. The training set from the Painting domain has 50,416 images while the test set has 21,850 images. (5) Infograph: infographic images with specific objects. The training set from the Infograph domain has 36,023 images while the test set has 15,582 images. We resize each image from all domains to be 96×96 resolution to save computational cost. We use the training set from the Real domain as $\mathcal{D}^{\text{tr}}$ and the test set from the Real domain as the source validation dataset. During training, we apply random horizontal flipping and random cropping with padding

data augmentations to the training images. We use the test sets from three domains Clipart, Sketch, and Painting as three different U_X for evaluation. So we evaluate three shifts: Real→Clipart (R→C), Real→Sketch (R→S), and Real→Painting (R→P). We use the remaining shift Real→Infograph (R→I) as a validation dataset for tuning the hyper-parameters.

Otto. The Otto Group Product Classification Challenge (Benjamin Bossan, 2015) is a tabular dataset hosted on Kaggle [1]. The task is to classify each product with 93 features into 9 categories. Each target category represents one of the most important product categories (like fashion, electronics, etc). It contains $61,878$ training data points. Since it only provides labels for the training data, we need to create the training, validation and test set. To create a test set that is from a different distribution than the training set, we apply the Local Outlier Factor (LOF) (Breunig et al., 2000), which is an unsupervised outlier detection method, on the Otto training data to identify a certain fraction (e.g., 0.2) of outliers as the test set. Specifically, we apply the *LocalOutlierFactor* function provided by scikit-learn (Pedregosa et al., 2011) on the training data with a contamination of 0.2 (contamination value determines the proportion of outliers in the data set) to identify the outliers. We identify $12,376$ outlier data points and use them as the test set U_X. We then randomly split the remaining data into a training set $\mathcal{D}^{\text{tr}}$ with $43,314$ data points and a source validation set with $6,188$ data points. We show that the test set indeed has a distribution shift compared to the source validation set, which causes the model trained on the training set to have a drop in performance (see Table 3 in Appendix F.1).

E.3 Details on Model Architectures and Training on Source Data

On all datasets, we use the following supervised training objective for training models on the source training set $\mathcal{D}^{\text{tr}}$:

$$\min_{\theta} \quad \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta) \tag{33}$$

where ℓ_{CE} is the cross-entropy loss and θ is the model parameters.

On MNIST→SVHN, we use the Convolutional Neural Network (CNN) (LeCun et al., 1989) consisting of four convolutional layers followed by two fully connected layers with batch normalization and dropout layers. We train the model on the training set of MNIST for 20 epochs using the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 10^{-3} and a batch size of 128.

On CIFAR-10→CINIC-10, we use the ResNet-20 network (He et al., 2016b). We train the model on the training set of CIFAR-10 for 200 epochs using the SGD optimizer with a learning rate of 0.1, a momentum of 0.9, and a batch size of 128. The learning rate is multiplied by 0.1 at the 80, 120, and 160 epochs, respectively, and is multiplied by 0.5 at the 180 epoch.

On the FMoW dataset, we use the DensetNet-121 network (Huang et al., 2017b) pre-trained on ImageNet. We train the model further for 50 epochs using the Adam optimizer with a learning rate of 10^{-4} and a batch size of 128.

On the Amazon Review dataset, we use the pre-trained RoBERTa base model (Liu et al., 2019) to extract the embedding of the input sentence for classification (i.e., RoBERTa's output for the [CLS] token) and then build an eight-layer fully connected neural network (also known as a multi-layer perceptron) with batch normalization, dropout layers and L2 regularization on top of the embedding. Note that we only update the parameters of the fully connected neural network without updating the parameters of the pre-trained RoBERTa base model during training (i.e., freeze the parameters of the RoBERTa base model during training). We train the model for 200 epochs using the Adam optimizer with a learning rate of 10^{-3} and a batch size of 128.

On the DomainNet dataset, we use the ResNet-50 network (He et al., 2016a) pre-trained on ImageNet. We train the model further on the training set from the Real domain for 50 epochs using the Adam optimizer with a learning rate of 10^{-4} and a batch size of 128.

On the Otto dataset, we use a six-layer fully connected neural network (also known as a multi-layer perceptron) with batch normalization, dropout layers and L2 regularization. We train the model on the created training set for 200 epochs using the Adam optimizer with a learning rate of 10^{-3} and a batch size of 128.

[1] URL: `https://kaggle.com/competitions/otto-group-product-classification-challenge`

E.4 ACTIVE LEARNING HYPER-PARAMETERS

During the active learning process, we fine-tune the model on the selected labeled test data. During fine-tuning, we don't apply any data augmentation to the test data. We use the same fine-tuning hyper-parameters for different methods to ensure a fair comparison. The optimizer used is the same as that in the source training stage (described in Appendix E.3). On MNIST→SVHN, we use a learning rate of 10^{-3}; On CIFAR-10→CINIC-10, we use a learning rate of 5×10^{-3}; On FMoW, we use a learning rate of 10^{-4}; On Amazon Review, we use a learning rate of 10^{-3}; On DomainNet, we use a learning rate of 10^{-4}; On Otto, we use a learning rate of 10^{-3}. On all datasets, we fine-tune the model for at least 50 epochs and up to 200 epochs with a batch size of 128 and early stopping using 10 patient epochs.

F ADDITIONAL EXPERIMENTAL RESULTS

F.1 EVALUATE SOURCE-TRAINED MODELS

In this section, we evaluate the accuracy of the source-trained models on the source validation dataset and the target test dataset U_X. The models are trained on the source training set $\mathcal{D}^{\text{tr}}$ (refer to Appendix E.3 for the details of source training). The source validation data are randomly sampled from the training data distribution while the target test data are sampled from a different distribution than the training data distribution. The results in Table 3 show that the models trained on $\mathcal{D}^{\text{tr}}$ always suffer a drop in accuracy when evaluating them on the target test dataset U_X.

Dataset	Source Accuracy	Target Accuracy
MNIST→SVHN	99.40	24.68
CIFAR-10→CINIC-10	90.46	71.05
FMoW	46.25	38.01
Amazon Review	65.39	61.40
DomainNet (R→C)	63.45	33.37
DomainNet (R→P)	63.45	26.29
DomainNet (R→S)	63.45	16.00
DomainNet (R→I)	63.45	8.11
Otto	80.72	66.09

Table 3: Results of evaluating the accuracy of the source-trained models on the source validation dataset and the target test dataset U_X. All numbers are percentages.

F.2 COMPLETE EVALUATION RESULTS

We give complete experimental results for the baselines and the proposed method ASPEST on all datasets in this section. We repeat each experiment three times with different random seeds and report the mean and standard deviation (std) values. These results are shown in Table 4 (MNIST→SVHN), Table 5 (CIFAR-10→CINIC-10), Table 6 (FMoW), Table 7 (Amazon Review), Table 8 (DomainNet R→C), Table 9 (DomainNet R→P), Table 10 (DomainNet R→S) and Table 11 (Otto). Our results show that the proposed method ASPEST consistently outperforms the baselines across different image, text and structured datasets.

F.3 EFFECT OF COMBINING SELECTIVE PREDICTION WITH ACTIVE LEARNING

Selective prediction without active learning corresponds to the case where the labeling budget $M = 0$ and the selected set $B^* = \emptyset$. To make fair comparisons with selective classification methods without active learning, we define a new coverage metric:

$$cov^*(f_s, \tau) = \mathbb{E}_{\mathbf{x} \sim U_X} \mathbb{I}[g(\mathbf{x}) \geq \tau \wedge \mathbf{x} \notin B^*] \tag{34}$$

The range of $cov^*(f_s, \tau)$ is $[0, 1 - \frac{M}{n}]$, where $M = |B^*|$ and $n = |U_X|$. If we use a larger labeling budget M for active learning, then the upper bound of $cov^*(f_s, \tau)$ will be smaller. Thus, in order to beat selective classification methods without active learning, active selective prediction methods need

Dataset	MNIST→SVHN								
Metric	$cov\|acc \geq 90\% \uparrow$			$acc\|cov \geq 90\% \uparrow$			AUC ↑		
Labeling Budget	100	500	1000	100	500	1000	100	500	1000
SR+Uniform	0.00±0.0	51.46±3.7	75.57±0.9	58.03±1.5	76.69±1.2	84.39±0.2	74.08±1.5	88.80±0.8	93.57±0.2
SR+Confidence	0.00±0.0	55.32±5.1	82.22±1.3	47.66±3.4	79.02±0.7	87.19±0.4	64.14±2.8	89.93±0.6	94.62±0.2
SR+Entropy	0.00±0.0	0.00±0.0	75.08±2.4	47.93±7.0	77.09±1.0	84.81±0.7	65.88±4.7	88.19±0.8	93.37±0.5
SR+Margin	0.00±0.0	63.60±2.7	82.19±0.3	61.39±0.5	80.96±0.9	86.97±0.2	76.79±0.5	91.24±0.5	94.82±0.1
SR+kCG	2.52±1.3	23.04±0.3	38.97±2.6	34.57±4.4	52.76±1.1	64.34±4.8	48.83±7.2	73.65±1.0	83.16±2.0
SR+CLUE	0.00±0.0	62.03±2.4	81.29±1.1	57.35±1.9	79.55±0.8	86.28±0.5	72.72±1.9	90.98±0.5	94.99±0.2
SR+BADGE	0.00±0.0	62.55±4.4	82.39±2.8	59.82±1.7	79.49±1.6	86.96±0.9	76.06±1.6	91.09±0.9	95.16±0.6
DE+Uniform	24.71±5.6	68.98±1.6	83.67±0.1	63.22±1.7	81.67±0.4	87.32±0.1	79.36±1.7	92.47±0.2	95.48±0.0
DE+Entropy	6.24±8.8	63.30±6.5	84.62±1.5	56.61±0.6	80.16±2.0	88.05±0.5	72.51±1.5	91.21±1.4	95.45±0.5
DE+Confidence	14.92±5.1	67.87±1.4	89.41±0.3	61.11±2.9	81.80±0.5	89.75±0.1	75.85±3.0	92.16±0.2	96.19±0.1
DE+Margin	21.59±3.8	77.84±2.8	92.75±0.3	62.88±1.2	85.11±1.1	91.17±0.1	78.59±1.4	94.31±0.6	97.00±0.0
DE+Avg-KLD	10.98±4.6	61.45±3.4	88.06±2.2	54.80±1.6	78.21±1.6	89.23±0.9	71.67±2.2	90.92±0.8	96.23±0.4
DE+CLUE	22.34±1.4	69.23±1.9	82.80±1.0	59.47±1.3	81.05±0.9	86.78±0.4	76.88±1.0	92.70±0.5	95.56±0.2
DE+BADGE	22.02±4.5	72.31±1.2	88.23±0.4	61.23±1.9	82.69±0.5	89.15±0.2	77.65±1.9	93.38±0.2	96.51±0.1
ASPEST (ours)	**52.10**±4.0	**89.22**±0.9	**98.70**±0.4	**76.10**±1.5	**89.62**±0.4	**93.92**±0.3	**88.84**±1.0	**96.62**±0.2	**98.06**±0.1

Table 4: Results of comparing ASPEST to the baselines on MNIST→SVHN. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	CIFAR-10→CINIC-10								
Metric	$cov\|acc \geq 90\% \uparrow$			$acc\|cov \geq 90\% \uparrow$			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
SR+Uniform	57.43±0.2	57.15±0.6	58.37±0.7	75.67±0.2	75.69±0.1	76.11±0.3	89.77±0.0	89.81±0.1	90.09±0.2
SR+Confidence	57.96±0.6	57.05±0.7	61.11±1.1	76.49±0.2	76.87±0.2	78.77±0.4	90.00±0.2	89.92±0.2	90.91±0.3
SR+Entropy	57.78±0.7	57.07±1.4	61.07±0.4	76.57±0.3	76.71±0.5	78.85±0.2	90.01±0.2	89.94±0.3	90.88±0.0
SR+Margin	57.72±0.8	57.98±0.7	61.71±0.2	76.24±0.2	76.90±0.2	78.42±0.2	89.95±0.2	90.14±0.1	91.02±0.0
SR+kCG	57.90±0.5	57.81±0.7	60.36±0.3	75.59±0.1	75.73±0.2	76.68±0.2	89.78±0.1	89.79±0.2	90.41±0.2
SR+CLUE	57.29±0.5	58.89±0.5	62.28±0.2	75.74±0.2	76.68±0.3	78.10±0.2	89.67±0.2	90.15±0.1	91.03±0.1
SR+BADGE	58.58±0.6	58.63±0.3	61.95±0.4	76.33±0.5	76.58±0.1	78.26±0.2	90.05±0.2	90.16±0.1	90.99±0.0
DE+Uniform	58.06±0.3	58.72±0.1	59.54±0.3	76.65±0.1	77.06±0.2	77.46±0.1	90.26±0.1	90.45±0.1	90.73±0.1
DE+Entropy	58.91±0.6	60.96±0.2	63.85±0.2	77.66±0.1	79.14±0.1	80.82±0.2	90.55±0.1	91.16±0.1	91.89±0.0
DE+Confidence	58.53±0.3	61.03±0.6	64.42±0.2	77.73±0.2	79.00±0.1	80.87±0.0	90.53±0.0	91.11±0.1	91.96±0.0
DE+Margin	58.76±0.5	61.60±0.5	64.92±0.5	77.61±0.2	78.91±0.1	80.59±0.1	90.56±0.1	91.11±0.1	91.98±0.1
DE+Avg-KLD	59.99±0.6	62.05±0.3	65.02±0.5	77.84±0.1	79.15±0.0	81.04±0.1	90.74±0.1	91.30±0.1	92.10±0.1
DE+CLUE	59.27±0.1	61.16±0.4	64.42±0.0	77.19±0.1	78.37±0.2	79.44±0.1	90.44±0.1	91.03±0.1	91.74±0.0
DE+BADGE	59.37±0.4	61.61±0.1	64.53±0.4	77.13±0.1	78.33±0.2	79.44±0.3	90.49±0.1	91.12±0.0	91.78±0.1
ASPEST (ours)	**60.38**±0.3	**63.34**±0.2	**66.81**±0.3	**78.23**±0.1	**79.49**±0.1	**81.25**±0.1	**90.95**±0.0	**91.60**±0.0	**92.33**±0.1

Table 5: Results of comparing ASPEST to the baselines on CIFAR-10→CINIC-10. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

to use a small labeling budget to achieve significant accuracy and coverage improvement. We still use the accuracy metric defined in (3). We then define a new maximum accuracy at a target coverage t_c as:

$$\max_{\tau} \quad acc(f_s, \tau), \quad s.t. \quad cov^*(f_s, \tau) \geq t_c \tag{35}$$

We denote this metric as $acc|cov^* \geq t_c$.

We define a new maximum coverage at a target accuracy t_a metric as:

$$\max_{\tau} \quad cov^*(f_s, \tau), \quad s.t. \quad acc(f_s, \tau) \geq t_a \tag{36}$$

We denote this metric as $cov^*|acc \geq t_a$.

The results under these new metrics are shown in Table 12 (MNIST→SVHN), Table 13 (CIFAR-10→CINIC-10 and Otto), Table 14 (FMoW and Amazon Review) and Table 15 (DomainNet). From the results, we can see that combining selective prediction with active learning can significantly improve the accuracy and coverage metrics, even with small labeling budgets.

F.4 Ablation Study for ASPEST

In this section, we perform ablation experiments on MNIST→SVHN and DomainNet R→C to analyze the effect of ensembling with checkpoints and self-training in ASPEST. The results are

Dataset	FMoW								
Metric	$cov\|acc \geq 70\% \uparrow$			$acc\|cov \geq 70\% \uparrow$			AUC $\uparrow$		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
SR+Uniform	38.50±0.7	42.00±0.5	52.34±1.1	51.76±0.7	54.27±0.2	60.31±0.7	65.75±0.4	67.67±0.3	72.73±0.3
SR+Confidence	37.34±0.3	42.28±1.2	53.72±0.7	52.24±0.1	55.52±0.5	61.76±0.4	65.57±0.1	68.03±0.5	73.14±0.5
SR+Entropy	37.42±0.3	42.08±0.2	51.18±0.4	51.74±0.4	54.94±0.2	60.62±0.2	65.31±0.2	68.00±0.1	71.99±0.2
SR+Margin	38.40±1.4	44.67±0.7	55.68±1.5	52.88±0.3	56.66±0.4	62.98±0.7	66.11±0.6	69.12±0.4	73.86±0.5
SR+kCG	36.50±0.8	39.76±1.2	45.87±0.6	49.36±0.7	51.45±0.5	55.47±0.1	64.34±0.5	66.21±0.6	69.63±0.2
SR+CLUE	38.65±0.7	44.50±1.8	54.71±0.5	52.23±0.4	55.54±1.0	61.13±0.4	65.78±0.3	68.76±0.9	73.80±0.1
SR+BADGE	40.47±1.5	45.65±1.2	57.59±0.4	53.08±1.0	56.63±0.3	63.57±0.2	66.74±0.8	69.43±0.6	74.76±0.2
DE+Uniform	44.74±0.4	51.57±1.1	61.92±0.4	56.39±0.5	60.01±0.5	65.74±0.2	69.44±0.3	72.48±0.5	77.02±0.1
DE+Entropy	43.76±0.3	50.52±1.4	62.73±0.4	56.29±0.3	60.31±0.3	66.53±0.2	69.02±0.1	72.10±0.5	76.65±0.2
DE+Confidence	45.23±0.6	50.11±0.9	64.29±0.3	57.18±0.4	60.46±0.3	67.46±0.0	69.80±0.3	72.11±0.4	77.37±0.1
DE+Margin	46.35±0.6	54.79±1.3	69.70±0.8	57.84±0.3	62.43±0.5	69.87±0.4	70.18±0.3	73.62±0.3	78.88±0.4
DE+Avg-KLD	46.29±0.3	53.63±0.8	68.18±0.9	57.75±0.4	61.60±0.3	69.11±0.4	70.16±0.1	73.09±0.2	78.48±0.3
DE+CLUE	45.22±0.2	49.97±0.3	58.05±0.5	56.39±0.1	59.05±0.1	63.23±0.4	69.53±0.0	71.95±0.1	75.72±0.3
DE+BADGE	47.39±0.7	53.83±0.7	66.45±0.8	57.71±0.4	61.16±0.2	68.13±0.4	70.59±0.4	73.40±0.3	78.66±0.1
ASPEST (ours)	**53.05**±0.4	**59.86**±0.4	**76.52**±0.6	**61.18**±0.2	**65.18**±0.2	**72.86**±0.3	**71.12**±0.2	**74.25**±0.2	**79.93**±0.1

Table 6: Results of comparing ASPEST to the baselines on FMoW. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	Amazon Review								
Metric	$cov\|acc \geq 80\% \uparrow$			$acc\|cov \geq 80\% \uparrow$			AUC $\uparrow$		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
SR+Uniform	13.71±11.3	24.10±5.3	24.87±2.6	65.13±0.8	66.33±0.6	66.26±0.3	72.71±1.5	73.64±0.7	73.53±0.7
SR+Confidence	11.28±8.9	17.96±4.0	33.19±1.4	65.15±0.7	66.29±0.4	68.94±0.1	72.89±0.7	73.25±0.7	76.17±0.2
SR+Entropy	5.55±7.8	13.32±9.5	25.47±1.8	65.11±1.1	66.56±0.7	67.31±0.7	71.96±1.6	72.53±1.1	74.19±0.5
SR+Margin	14.48±10.9	22.61±4.2	28.35±6.1	65.75±0.5	66.31±0.4	68.15±0.4	73.25±1.0	73.65±0.5	75.17±0.8
SR+kCG	20.02±11.0	17.02±12.2	29.08±4.2	64.03±3.1	66.17±0.5	66.63±1.0	72.34±3.2	74.35±0.7	74.49±1.0
SR+CLUE	4.15±5.9	25.15±4.9	31.88±2.1	66.17±0.4	66.30±0.4	67.12±0.7	73.43±0.4	74.07±0.7	75.29±0.9
SR+BADGE	22.58±0.4	23.78±6.4	30.71±4.6	66.29±0.4	66.31±0.6	68.58±0.7	73.80±0.6	74.00±1.0	75.76±0.8
DE+Uniform	34.35±1.4	33.15±1.1	36.55±1.8	68.13±0.4	68.12±0.6	68.88±0.2	76.20±0.3	76.16±0.4	77.07±0.3
DE+Entropy	31.74±1.4	36.29±1.6	40.33±1.7	68.19±0.3	69.44±0.2	71.27±0.3	75.98±0.4	77.10±0.3	78.53±0.3
DE+Confidence	35.12±1.8	34.48±1.4	40.46±0.5	69.07±0.3	69.47±0.2	71.08±0.2	76.63±0.2	76.87±0.3	78.27±0.1
DE+Margin	33.42±1.3	35.03±1.3	41.20±0.4	68.45±0.3	69.30±0.2	70.88±0.1	76.18±0.2	76.91±0.3	78.31±0.1
DE+Avg-KLD	33.03±1.5	38.55±3.2	41.75±1.8	68.63±0.3	69.95±0.4	71.10±0.3	76.21±0.4	77.62±0.6	78.62±0.3
DE+CLUE	33.92±3.0	35.27±1.4	34.83±3.1	68.09±0.3	68.07±0.3	68.40±0.6	76.27±0.6	76.65±0.3	76.69±0.7
DE+BADGE	32.23±3.7	36.18±1.5	40.58±3.3	68.34±0.4	68.87±0.2	70.29±0.3	76.13±0.7	77.09±0.2	78.44±0.5
ASPEST (ours)	**38.44**±0.7	**40.96**±0.8	**45.77**±0.1	**69.31**±0.3	**70.17**±0.2	**71.60**±0.2	**77.69**±0.1	**78.35**±0.2	**79.51**±0.2

Table 7: Results of comparing ASPEST to the baselines on Amazon Review. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

shown in Table 16. From the results, we can see that for MNIST→SVHN, adding the checkpoint ensemble component alone (ASPEST w/o self-training) does not improve the performance over DE+Margin, whereas adding the self-training component alone (ASPEST w/o checkpoint ensemble)

Dataset	DomainNet R→C (easy)								
Metric	$cov\|acc \geq 80\% \uparrow$			$acc\|cov \geq 80\% \uparrow$			AUC $\uparrow$		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
SR+Uniform	25.56±0.6	27.68±0.8	29.86±0.0	43.63±0.4	45.57±0.3	47.27±0.4	63.31±0.4	65.11±0.5	66.70±0.2
SR+Confidence	25.96±0.2	27.80±1.2	32.51±1.3	44.90±0.8	47.26±0.4	52.04±0.8	64.20±0.6	65.88±0.6	69.70±0.7
SR+Entropy	25.44±1.0	27.79±0.4	33.51±1.1	44.46±0.7	46.96±0.3	52.25±0.5	63.52±0.6	65.72±0.2	70.03±0.5
SR+Margin	26.28±1.2	27.77±1.0	32.92±0.4	45.24±1.0	47.12±0.7	52.29±0.4	64.37±0.8	65.91±0.6	70.01±0.4
SR+kCG	21.12±0.3	21.79±0.4	23.43±0.5	39.19±0.1	40.59±0.4	41.11±0.3	58.88±0.0	60.11±0.4	60.89±0.1
SR+CLUE	27.17±0.8	29.78±0.8	34.82±0.6	44.57±0.7	46.79±0.1	49.70±0.3	64.38±0.6	66.47±0.3	69.59±0.1
SR+BADGE	27.78±0.8	30.78±0.6	36.00±0.6	45.36±0.6	48.43±0.6	53.00±0.4	64.90±0.5	67.56±0.4	71.39±0.4
DE+Uniform	30.82±0.8	33.05±0.4	36.80±0.2	48.19±0.3	50.09±0.3	52.98±0.5	67.60±0.4	69.31±0.3	71.64±0.4
DE+Entropy	29.13±0.9	34.07±0.3	40.82±0.3	48.67±0.4	51.66±0.2	57.81±0.2	67.48±0.3	70.05±0.2	74.64±0.2
DE+Confidence	29.90±0.8	33.73±0.2	40.80±0.2	48.60±0.3	52.03±0.3	58.43±0.1	67.45±0.3	70.19±0.2	74.80±0.1
DE+Margin	31.82±1.3	35.68±0.2	43.39±0.7	50.12±0.4	53.19±0.4	59.17±0.2	68.85±0.4	71.29±0.3	75.79±0.3
DE+Avg-KLD	32.23±0.2	36.09±0.6	44.00±0.5	49.81±0.3	53.38±0.3	58.93±0.1	68.73±0.2	71.40±0.2	75.73±0.2
DE+CLUE	30.80±0.3	33.04±0.4	35.52±0.2	48.56±0.3	49.91±0.3	51.40±0.2	67.82±0.2	69.10±0.2	70.62±0.2
DE+BADGE	30.16±1.3	36.18±0.3	43.34±0.3	49.78±0.3	53.26±0.1	58.65±0.4	68.46±0.3	71.35±0.2	75.37±0.3
ASPEST (ours)	**37.38**±0.1	**39.98**±0.3	**48.29**±1.0	**54.56**±0.3	**56.95**±0.1	**62.69**±0.2	**71.61**±0.2	**73.27**±0.2	**77.40**±0.4

Table 8: Results of comparing ASPEST to the baselines on DomainNet R→C. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	DomainNet R→P (medium)								
Metric	$cov\|acc \geq 70\% \uparrow$			$acc\|cov \geq 70\% \uparrow$			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
SR+Uniform	21.01±1.0	21.35±0.3	22.64±0.5	36.78±0.6	37.18±0.2	38.20±0.4	51.87±0.7	52.31±0.0	53.34±0.4
SR+Confidence	20.64±0.6	22.15±0.8	23.60±0.6	37.01±0.3	38.46±0.7	40.23±0.4	51.77±0.3	53.33±0.8	54.80±0.5
SR+Entropy	20.76±0.7	22.11±0.3	23.56±0.3	37.09±0.2	38.38±0.3	40.30±0.1	51.86±0.4	53.29±0.3	54.81±0.2
SR+Margin	21.43±0.4	23.29±0.3	24.70±1.0	37.21±0.2	39.15±0.4	40.81±0.4	52.33±0.1	54.09±0.3	55.70±0.4
SR+kCG	17.33±0.4	17.62±0.2	18.49±0.2	33.97±0.3	34.12±0.1	34.36±0.1	48.61±0.5	48.65±0.2	49.25±0.2
SR+CLUE	21.15±0.6	22.49±0.5	24.84±0.7	36.96±0.2	37.93±0.5	39.31±0.4	51.97±0.4	53.20±0.5	54.84±0.5
SR+BADGE	20.07±0.3	22.21±0.5	24.92±0.2	36.10±0.1	38.11±0.4	40.40±0.5	50.99±0.0	53.10±0.4	55.40±0.4
DE+Uniform	25.42±0.2	26.38±0.2	28.83±0.3	40.83±0.1	41.66±0.2	43.93±0.2	55.86±0.1	56.62±0.1	58.80±0.2
DE+Entropy	25.74±0.4	27.11±0.4	30.39±0.1	41.34±0.1	42.92±0.3	45.92±0.3	56.06±0.2	57.51±0.3	60.10±0.2
DE+Confidence	25.69±0.4	27.38±0.7	30.47±0.1	41.45±0.2	43.12±0.3	45.88±0.1	56.13±0.2	57.68±0.3	60.20±0.2
DE+Margin	25.78±0.3	27.88±0.5	31.03±0.4	41.26±0.2	43.13±0.3	46.23±0.4	56.23±0.2	57.90±0.3	60.49±0.3
DE+Avg-KLD	26.30±0.7	28.00±0.1	31.97±0.2	41.80±0.3	43.17±0.1	46.32±0.2	56.65±0.3	57.99±0.1	60.82±0.2
DE+CLUE	25.38±0.6	26.65±0.4	27.89±0.1	40.86±0.3	41.62±0.2	42.46±0.1	55.79±0.4	56.65±0.2	57.71±0.1
DE+BADGE	26.27±0.7	27.69±0.1	31.84±0.2	42.02±0.6	43.41±0.2	46.37±0.1	56.67±0.5	58.03±0.1	60.84±0.1
ASPEST (ours)	**29.69**±0.1	**32.50**±0.3	**35.46**±0.6	**44.96**±0.1	**46.77**±0.2	**49.42**±0.1	**58.74**±0.0	**60.36**±0.0	**62.84**±0.2

Table 9: Results of comparing ASPEST to the baselines on DomainNet R→P. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	DomainNet R→S (hard)								
Metric	$cov\|acc \geq 70\% \uparrow$			$acc\|cov \geq 70\% \uparrow$			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
SR+Uniform	12.12±0.7	12.42±0.4	15.88±0.2	27.01±0.6	27.74±0.3	31.29±0.3	41.12±0.8	41.89±0.2	46.17±0.3
SR+Confidence	11.06±1.1	11.48±0.5	14.49±1.5	26.53±1.4	27.98±0.2	31.31±0.7	40.26±1.6	41.65±0.2	45.46±1.1
SR+Entropy	10.91±0.3	12.45±0.6	14.65±0.6	26.84±0.5	28.72±0.5	31.07±0.6	40.47±0.6	42.61±0.8	45.31±0.4
SR+Margin	12.23±0.4	13.06±0.4	15.31±0.4	27.87±0.2	29.19±0.4	31.51±0.8	41.91±0.3	43.22±0.4	45.97±0.8
SR+kCG	9.03±0.2	9.76±0.2	11.41±0.2	23.32±0.4	24.06±0.4	25.68±0.4	36.63±0.3	37.57±0.4	39.80±0.3
SR+CLUE	12.39±0.3	14.17±1.0	15.80±0.8	27.82±0.4	29.68±0.4	30.62±0.8	42.00±0.4	44.19±0.7	45.58±0.9
SR+BADGE	12.18±0.9	13.13±1.0	15.83±0.7	27.68±1.0	28.96±0.7	32.00±0.4	41.72±1.1	43.28±0.9	46.60±0.6
DE+Uniform	15.91±0.5	17.55±0.4	21.33±0.3	31.37±0.5	32.57±0.4	36.12±0.2	46.28±0.5	47.79±0.4	51.64±0.2
DE+Entropy	13.70±0.3	16.31±0.5	19.58±0.4	30.38±0.4	32.45±0.2	36.18±0.2	44.79±0.5	47.15±0.2	50.87±0.3
DE+Confidence	13.73±0.2	16.21±0.2	19.22±0.4	30.55±0.3	33.02±0.1	36.29±0.5	45.05±0.3	47.59±0.0	50.84±0.4
DE+Margin	14.99±0.2	17.45±0.4	21.74±0.7	31.67±0.5	33.51±0.5	37.88±0.3	46.38±0.5	48.44±0.5	52.78±0.4
DE+Avg-KLD	15.75±0.5	18.14±0.7	22.15±0.3	31.36±0.2	33.79±0.2	37.96±0.2	46.29±0.1	48.77±0.3	53.02±0.3
DE+CLUE	14.76±0.5	17.38±0.1	19.75±0.4	31.05±0.4	32.58±0.2	34.61±0.4	45.80±0.3	47.74±0.1	50.09±0.2
DE+BADGE	14.97±0.1	17.49±0.3	21.71±0.3	31.35±0.2	33.46±0.1	37.35±0.3	46.03±0.1	48.31±0.1	52.33±0.2
ASPEST (ours)	**17.86**±0.4	**20.42**±0.4	**25.87**±0.4	**35.17**±0.1	**37.28**±0.3	**41.46**±0.2	**49.62**±0.1	**51.61**±0.4	**55.90**±0.2

Table 10: Results of comparing ASPEST to the baselines on DomainNet R→S. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

can significantly improve the performance. For DomainNet R→C, both checkpoint ensemble and self-training components have positive contributions. For both cases, ASPEST (with both self-training and checkpoint ensemble) achieves much better results than DE+Margin or applying those components alone. We also show that the performance is not highly sensitive to η, while typically setting larger η (e.g. $\eta = 0.9$) yields better results.

F.5 Effect of joint training

In the problem setup, we assume that we have access to the training dataset $\mathcal{D}^{\text{tr}}$ and can use joint training to improve the selective prediction performance. In this section, we perform experiments to study the effect of joint training and the effect of the loss coefficient λ when performing joint training. We consider three active selective prediction methods: SR+margin (Algorithm 2 with margin sampling), DE+margin (Algorithm 3 with margin sampling), and ASPEST (Algorithm 1). We consider $\lambda \in \{0, 0.5, 1.0, 2.0\}$. When $\lambda = 0$, we don't use joint training; when $\lambda > 0$, we use joint training. The results are shown in Table 17. From the results, we can see that using joint training (i.e., when $\lambda > 0$) can improve performance, especially when the labeling budget is small. Also, setting a too large value for λ (e.g., $\lambda = 2$) will lead to worse performance. Setting $\lambda = 0.5$ or 1 usually leads to better performance. In our experiments, we simply set $\lambda = 1$ by default.

Dataset	Otto								
Metric	$cov\|acc \geq 80\% \uparrow$			$acc\|cov \geq 80\% \uparrow$			AUC $\uparrow$		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
SR+Uniform	63.58±0.7	64.06±0.4	67.49±0.9	73.56±0.3	73.57±0.6	75.21±0.2	84.46±0.2	84.61±0.3	85.72±0.2
SR+Confidence	69.63±1.7	73.41±0.6	84.19±0.5	75.96±0.5	77.57±0.2	81.39±0.2	85.91±0.3	86.86±0.1	88.93±0.1
SR+Entropy	67.79±0.8	73.83±1.0	83.12±0.7	75.43±0.4	77.91±0.3	81.07±0.2	85.41±0.3	86.94±0.2	88.86±0.1
SR+Margin	68.10±0.1	74.10±0.4	82.53±0.2	75.52±0.0	77.66±0.1	80.93±0.1	85.56±0.1	86.99±0.1	88.83±0.1
SR+kCG	64.84±0.7	62.90±1.1	59.85±1.0	73.75±0.3	73.03±0.2	71.90±0.3	85.08±0.2	84.67±0.2	83.79±0.3
SR+CLUE	68.21±1.2	70.85±0.6	78.26±0.9	75.26±0.5	76.32±0.2	79.30±0.3	85.82±0.3	86.69±0.2	88.53±0.2
SR+BADGE	67.23±1.0	73.52±0.2	83.17±0.4	74.74±0.3	77.43±0.2	81.20±0.2	85.41±0.3	87.10±0.2	89.25±0.1
DE+Uniform	70.74±0.5	72.20±0.6	75.58±0.5	76.40±0.1	77.06±0.2	78.35±0.2	86.78±0.1	87.26±0.1	88.11±0.1
DE+Entropy	75.71±0.3	80.91±0.2	92.62±0.2	78.44±0.1	80.29±0.1	84.05±0.1	87.87±0.1	88.77±0.1	90.99±0.1
DE+Confidence	75.52±0.2	81.69±0.7	92.15±0.9	78.28±0.1	80.49±0.2	83.83±0.1	87.84±0.1	89.05±0.1	90.98±0.1
DE+Margin	75.49±0.8	81.36±0.8	92.49±0.4	78.41±0.3	80.50±0.2	84.06±0.2	87.89±0.2	89.10±0.2	90.95±0.2
DE+Avg-KLD	75.91±0.2	80.97±0.5	91.94±0.8	78.50±0.1	80.33±0.2	83.80±0.2	87.89±0.0	89.06±0.1	90.98±0.1
DE+CLUE	69.66±0.5	70.52±0.1	70.17±0.4	76.09±0.3	76.32±0.1	76.31±0.2	86.67±0.1	87.11±0.0	87.06±0.1
DE+BADGE	73.23±0.2	77.89±0.6	86.32±0.5	77.33±0.1	79.21±0.3	82.32±0.2	87.55±0.1	88.75±0.1	90.58±0.0
ASPEST (ours)	**77.85**±0.2	**84.20**±0.6	**94.26**±0.6	**79.28**±0.1	**81.40**±0.1	**84.62**±0.1	**88.28**±0.1	**89.61**±0.1	**91.49**±0.0

Table 11: Results of comparing ASPEST to the baselines on Otto. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	MNIST→SVHN	
Metric	$cov^*\|acc \geq 90\% \uparrow$	$acc\|cov^* \geq 90\% \uparrow$
SR (w/o active learning)	0.08±0.0	25.80±0.0
SR+Margin (M=500)	62.38±2.7	80.21±0.9
SR+Margin (M=1000)	79.04±0.2	85.36±0.3
DE (w/o active learning)	0.12±0.1	28.17±0.5
DE+Margin (M=500)	76.35±2.7	84.34±1.1
DE+Margin (M=1000)	89.19±0.3	89.59±0.1
ASPEST (M=500)	87.51±0.9	88.88±0.4
ASPEST (M=1000)	**94.91**±0.4	**92.44**±0.2

Table 12: Results on MNIST→SVHN to describe the effect of combining selective prediction with active learning. "w/o" means "without". The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

F.6 Effect of the number of rounds T

In this section, we study the effect of the number of rounds T in active learning. From the results in Table 18, we can see that larger T usually leads to better performance, and the proposed method ASPEST has more improvement as we increase T compared to SR+Margin and DE+Margin. Also, when T is large enough, the improvement becomes minor (or can even be worse). Considering that in practice, we might not be able to set a large T due to resource constraints, we thus set $T = 10$ by default.

Dataset	CIFAR-10→CINIC-10		Otto	
Metric	$cov^*\|acc \geq 90\% \uparrow$	$acc\|cov^* \geq 90\% \uparrow$	$cov^*\|acc \geq 80\% \uparrow$	$acc\|cov^* \geq 80\% \uparrow$
SR (w/o active learning)	57.43±0.0	75.62±0.0	62.90±0.0	73.13±0.0
SR+Margin (M=500)	56.76±0.8	75.61±0.2	65.34±0.1	74.25±0.1
SR+Margin (M=1000)	56.04±0.7	75.70±0.1	68.11±0.4	74.99±0.2
DE (w/o active learning)	56.64±0.2	75.83±0.1	67.69±0.4	75.41±0.2
DE+Margin (M=500)	57.78±0.5	76.96±0.2	72.44±0.7	77.18±0.3
DE+Margin (M=1000)	59.55±0.5	77.59±0.1	74.78±0.7	78.19±0.2
ASPEST (M=500)	59.37±0.3	77.60±0.1	74.71±0.2	77.99±0.2
ASPEST (M=1000)	**61.23**±0.2	**78.16**±0.1	**77.40**±0.5	**79.05**±0.2

Table 13: Results on CIFAR-10→CINIC-10 and Otto for studying the effect of combining selective prediction with active learning. "w/o" means "without". The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	FMoW		Amazon Review	
Metric	$cov^*\|acc \geq 70\% \uparrow$	$acc\|cov^* \geq 70\% \uparrow$	$cov^*\|acc \geq 80\% \uparrow$	$acc\|cov^* \geq 80\% \uparrow$
SR (w/o active learning)	32.39±0.0	48.15±0.0	26.79±0.0	65.64±0.0
SR+Margin (M=500)	37.54±1.3	52.19±0.3	14.16±10.6	65.38±0.4
SR+Margin (M=1000)	42.65±0.7	55.30±0.5	21.60±4.0	65.68±0.4
DE (w/o active learning)	37.58±0.3	52.01±0.1	35.81±1.9	68.41±0.2
DE+Margin (M=500)	45.30±0.6	57.09±0.3	32.68±1.2	68.10±0.3
DE+Margin (M=1000)	52.32±1.2	60.96±0.4	33.47±1.2	68.54±0.2
ASPEST (M=500)	51.85±0.4	60.43±0.2	37.59±0.6	68.91±0.2
ASPEST (M=1000)	**57.15**±0.4	**63.71**±0.2	**39.14**±0.8	**69.31**±0.2

Table 14: Results on FMoW and Amazon Review for studying the effect of combining selective prediction with active learning. "w/o" means "without". The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	DomainNet R→C (easy)		DomainNet R→P (medium)		DomainNet R→S (hard)	
Metric	$cov^*\|acc \geq 80\% \uparrow$	$acc\|cov^* \geq 80\% \uparrow$	$cov^*\|acc \geq 70\% \uparrow$	$acc\|cov^* \geq 70\% \uparrow$	$cov^*\|acc \geq 70\% \uparrow$	$acc\|cov^* \geq 70\% \uparrow$
SR (w/o active learning)	21.50±0.0	40.16±0.0	18.16±0.0	34.74±0.0	7.16±0.0	21.24±0.0
SR+Margin (M=500)	25.38±1.1	44.09±0.9	20.94±0.4	36.65±0.1	11.94±0.4	27.35±0.2
SR+Margin (M=1000)	25.87±1.0	44.70±0.7	22.22±0.3	37.91±0.4	12.43±0.4	28.19±0.4
DE (w/o active learning)	26.15±0.2	44.51±0.1	22.44±0.2	39.06±0.1	9.90±0.4	25.37±0.0
DE+Margin (M=500)	30.73±1.2	48.85±0.4	25.19±0.3	40.59±0.1	14.63±0.2	31.11±0.5
DE+Margin (M=1000)	33.24±0.2	50.46±0.4	26.60±0.5	41.73±0.3	16.62±0.4	32.30±0.5
ASPEST (M=500)	36.10±0.1	53.22±0.3	29.01±0.1	44.26±0.1	17.43±0.4	34.55±0.1
ASPEST (M=1000)	**37.24**±0.3	**54.03**±0.1	**31.01**±0.3	**45.31**±0.1	**19.45**±0.3	**35.96**±0.3

Table 15: Results on DomainNet R→C, R→P and R→S for studying the effect of combining selective prediction with active learning. "w/o" means "without". The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

F.7 Effect of the number of models N in the ensemble

In this section, we study the effect of the number of models N in the ensemble for DE+Margin and ASPEST. The results in Table 19 show that larger N usually leads to better results. However, larger N also means a larger computational cost. In our experiments, we simply set $N = 5$ by default.

F.8 Effect of the upper bound in pseudo-labeled set construction

When constructing the pseudo-labeled set R using Eq. (12), we exclude those test data points with confidence equal to 1. In this section, we study whether setting such an upper bound can improve performance. The results in Table 20 show that when the labeling budget is small, setting such an upper bound can improve performance significantly. However, when the labeling budget is large,

Dataset	MNIST→SVHN		DomainNet R→C (easy)	
Metric	AUC ↑		AUC ↑	
Labeling Budget	100	500	500	1000
DE+Margin	78.59±1.4	94.31±0.6	68.85±0.4	71.29±0.3
ASPEST w/o self-training	78.09±1.3	94.25±0.4	69.59±0.2	72.45±0.1
ASPEST w/o checkpoint ensemble	83.78±2.9	96.54±0.2	69.94±0.1	72.20±0.4
ASPEST (η=0.1)	83.77±1.7	96.01±0.4	70.35±0.2	72.89±0.4
ASPEST (η=0.5)	83.99±1.3	96.24±0.2	70.92±0.3	73.37±0.1
ASPEST (η=0.6)	85.17±1.3	96.24±0.2	70.96±0.2	73.05±0.1
ASPEST (η=0.8)	85.40±2.3	96.74±0.1	71.05±0.2	72.99±0.3
ASPEST (η=0.9)	**88.84**±1.0	96.62±0.2	**71.61**±0.2	73.27±0.2
ASPEST (η=0.95)	87.67±1.3	**96.74**±0.1	71.03±0.3	**73.38**±0.2

Table 16: Ablation study results for the proposed method ASPEST. "w/o" means "without". The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	MNIST→SVHN		DomainNet R→C (easy)	
Metric	AUC ↑		AUC ↑	
Labeling Budget	100	500	500	1000
SR+Margin ($\lambda = 0$)	71.90±3.1	90.56±0.8	60.05±0.9	60.34±1.2
SR+Margin ($\lambda = 0.5$)	75.54±1.7	91.43±0.5	64.99±0.7	66.81±0.5
SR+Margin ($\lambda = 1$)	76.79±0.5	91.24±0.5	64.37±0.8	65.91±0.6
SR+Margin ($\lambda = 2$)	72.71±2.5	90.80±0.3	64.17±0.3	66.21±0.2
DE+Margin ($\lambda = 0$)	77.12±0.5	94.26±0.5	66.86±0.5	69.29±0.6
DE+Margin ($\lambda = 0.5$)	79.35±1.4	94.22±0.2	69.28±0.3	71.60±0.2
DE+Margin ($\lambda = 1$)	78.59±1.4	94.31±0.6	68.85±0.4	71.29±0.3
DE+Margin ($\lambda = 2$)	77.64±2.3	93.81±0.4	68.54±0.1	71.28±0.2
ASPEST ($\lambda = 0$)	84.48±2.5	96.99±0.2	68.61±1.2	73.21±1.2
ASPEST ($\lambda = 0.5$)	86.46±3.1	97.01±0.0	71.53±0.1	73.69±0.1
ASPEST ($\lambda = 1$)	88.84±1.0	96.62±0.2	71.61±0.2	73.27±0.2
ASPEST ($\lambda = 2$)	85.46±1.7	96.43±0.1	70.54±0.3	73.02±0.1

Table 17: Ablation study results for the effect of using joint training and the effect of the loss coefficient λ. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages.

Dataset	MNIST→SVHN		DomainNet R→C (easy)	
Metric	AUC ↑		AUC ↑	
Labeling Budget	100	500	500	1000
SR+Margin (T=1)	63.10±2.7	75.42±3.6	65.16±0.4	66.76±0.3
SR+Margin (T=2)	68.09±3.1	87.45±1.6	64.64±0.8	66.91±0.1
SR+Margin (T=5)	74.87±1.7	91.32±0.5	64.35±0.2	66.76±0.3
SR+Margin (T=10)	76.79±0.5	91.24±0.5	64.37±0.8	65.91±0.6
SR+Margin (T=20)	72.81±1.5	90.34±1.3	63.65±0.6	66.08±0.4
DE+Margin (T=1)	69.85±0.5	82.74±2.1	68.39±0.2	70.55±0.0
DE+Margin (T=2)	75.25±1.0	90.90±1.0	68.79±0.2	70.95±0.5
DE+Margin (T=5)	78.41±0.2	93.26±0.3	68.80±0.2	71.21±0.2
DE+Margin (T=10)	78.59±1.4	94.31±0.6	68.85±0.4	71.29±0.3
DE+Margin (T=20)	76.84±0.4	94.67±0.2	68.50±0.5	71.39±0.2
ASPEST (T=1)	62.53±1.0	80.72±1.5	69.44±0.1	71.79±0.2
ASPEST (T=2)	75.08±1.4	89.70±0.7	70.68±0.2	72.56±0.3
ASPEST (T=5)	81.57±1.8	95.43±0.1	71.23±0.1	73.19±0.1
ASPEST (T=10)	88.84±1.0	96.62±0.2	71.61±0.2	73.27±0.2
ASPEST (T=20)	91.26±0.9	97.32±0.1	70.57±0.4	73.32±0.3

Table 18: Ablation study results for the effect of the number of rounds T. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages.

setting such an upper bound may not improve the performance. Since we focus on the low labeling budget region, we decide to set such an upper bound for the proposed ASPEST method.

F.9 Ensemble accuracy after each round of active learning

We evaluate the accuracy of the ensemble model f_t in the ASPEST algorithm after the t-th round of active learning. Recall that f_t contains N models $f_t^1, \ldots, f_t^N$ and $f_t(\mathbf{x}) = \arg\max_{k \in \mathcal{Y}} \frac{1}{N} \sum_{j=1}^{N} f_t^j(\mathbf{x} \mid k)$. The results in Table 21 show that after each round of active learning, the accuracy of the ensemble model will be improved significantly.

Dataset	MNIST→SVHN		DomainNet R→C (easy)	
Metric	AUC ↑		AUC ↑	
Labeling Budget	100	500	500	1000
DE+Margin (N=2)	67.41±3.9	91.20±0.8	65.82±0.5	67.72±0.4
DE+Margin (N=3)	77.53±1.5	93.41±0.1	67.54±0.4	69.61±0.2
DE+Margin (N=4)	74.46±2.7	93.65±0.3	68.09±0.2	70.65±0.3
DE+Margin (N=5)	78.59±1.4	94.31±0.6	68.85±0.4	71.29±0.3
DE+Margin (N=6)	79.34±0.7	94.40±0.1	68.63±0.2	71.65±0.3
DE+Margin (N=7)	80.30±1.5	93.97±0.2	69.41±0.1	71.78±0.3
DE+Margin (N=8)	78.91±1.5	94.52±0.2	69.00±0.0	71.88±0.4
ASPEST (N=2)	80.38±1.2	96.26±0.0	69.14±0.3	71.36±0.3
ASPEST (N=3)	84.86±1.0	96.60±0.2	69.91±0.2	72.25±0.2
ASPEST (N=4)	84.94±0.3	96.76±0.1	70.68±0.2	73.09±0.2
ASPEST (N=5)	88.84±1.0	96.62±0.2	71.61±0.2	73.27±0.2
ASPEST (N=6)	84.51±0.5	96.66±0.2	71.20±0.2	73.42±0.3
ASPEST (N=7)	86.70±2.3	96.90±0.2	71.16±0.2	73.50±0.1
ASPEST (N=8)	88.59±0.9	97.01±0.1	71.62±0.3	73.76±0.2

Table 19: Ablation study results for the effect of the number of models N in the ensemble. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages.

Dataset	MNIST→SVHN		DomainNet R→C (easy)	
Metric	AUC ↑		AUC ↑	
Labeling Budget	100	500	500	1000
ASPEST without upper bound	86.95±1.4	96.59±0.1	71.39±0.1	**73.52**±0.2
ASPEST	**88.84**±1.0	**96.62**±0.2	**71.61**±0.2	73.27±0.2

Table 20: Ablation study results for the effect of setting an upper bound when constructing the pseudo-labeled set R in ASPEST. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

F.10 Empirical analysis for checkpoint ensemble

In this section, we analyze why the proposed checkpoint ensemble could improve selective prediction performance. We postulate the rationales: (1) the checkpoint ensemble can help with generalization; (2) the checkpoint ensemble can help with reducing overconfident wrong predictions.

Metric	Ensemble Test Accuracy			
Dataset	MNIST→SVHN		DomainNet R→C (easy)	
Labeling Budget	100	500	500	1000
Round 0	24.67	24.87	37.33	37.46
Round 1	24.91	43.80	39.61	39.67
Round 2	37.75	54.91	41.15	41.55
Round 3	45.62	64.15	41.97	43.24
Round 4	50.94	71.65	42.57	45.09
Round 5	56.75	77.23	43.85	45.62
Round 6	59.82	79.97	44.20	46.60
Round 7	63.10	81.43	45.02	47.51
Round 8	67.49	82.78	45.17	48.59
Round 9	69.93	84.70	45.80	48.66
Round 10	71.14	85.48	46.36	49.70

Table 21: Ensemble test accuracy of ASPEST after each round of active learning. All numbers are percentages.

Regarding (1), when fine-tuning the model on the small set of selected labeled test data, we hope that the fine-tuned model could generalize to remaining unlabeled test data. However, since the selected test set is small, we might have an overfitting issue. So possibly some intermediate checkpoints along the training path achieve better generalization than the end checkpoint. By using checkpoint ensemble, we might get an ensemble that achieves better generalization to remaining unlabeled test data.

Regarding (2), when fine-tuning the model on the small set of selected labeled test data, the model can get increasingly confident on the test data. Since there exist high-confidence mis-classified test points, incorporating intermediate checkpoints along the training path into the ensemble can reduce the average confidence of the ensemble on those mis-classified test points. By using checkpoint ensemble, we might get an ensemble that has better confidence calibration on the test data.

We perform experiments on the image dataset MNIST→SVHN, the text dataset Amazon Review and the tabular dataset Otto to verify these two hypotheses. We evaluate the Accuracy (Acc), the Expected Calibration Error (ECE) and the area under the accuracy-coverage curve (AUC) metrics of the checkpoints during fine-tuning and the checkpoint ensemble constructed after fine-tuning on the target test dataset. We employ one-round active learning with a labeling budget of 100 samples. We use the margin sampling method for sample selection and fine-tune a single model on the selected labeled test data for 100 epochs. The results in Table 22 show that in the fine-tuning path, different checkpoints have different target test accuracy and the end checkpoint may not have the optimal target test accuracy. The checkpoint ensemble can have better target test accuracy than the end checkpoint. Also, in the fine-tuning path, different checkpoints have different confidence calibration (measured by the metric ECE) on the target test data and the end checkpoint may not have the optimal confidence calibration. The checkpoint ensemble can have better confidence calibration than the end checkpoint. Furthermore, in the fine-tuning path, different checkpoints have different selective prediction performance (measured by the metric AUC) on the target test data and the end checkpoint may not have the optimal selective prediction performance. The checkpoint ensemble can have better selective prediction performance than the end checkpoint.

Dataset	MNIST→SVHN			Amazon Review			Otto		
Metric	Acc↑	ECE↓	AUC↑	Acc↑	ECE↓	AUC↑	Acc↑	ECE↓	AUC↑
Checkpoint at epoch 5	31.52	11.17	42.44	60.88	20.69	72.83	67.53	31.78	85.61
Checkpoint at epoch 10	37.23	8.55	51.75	61.51	19.84	74.19	67.39	31.97	85.64
Checkpoint at epoch 15	38.84	8.87	55.67	61.58	20.38	74.91	67.27	31.78	85.55
Checkpoint at epoch 20	41.28	13.65	57.53	61.94	21.41	75.21	67.13	31.81	85.49
Checkpoint at epoch 25	42.81	18.90	59.40	61.76	22.65	75.12	67.28	31.56	85.55
Checkpoint at epoch 30	43.69	22.14	61.08	61.64	23.17	74.51	67.53	31.30	85.61
Checkpoint at epoch 35	44.78	25.38	62.57	61.75	23.39	74.45	67.59	31.22	85.62
Checkpoint at epoch 40	44.82	27.43	62.36	61.81	23.81	74.38	67.57	31.52	85.68
Checkpoint at epoch 45	45.05	29.76	61.67	61.88	23.86	74.24	67.66	31.60	85.70
Checkpoint at epoch 50	44.36	31.41	61.22	62.08	23.63	74.51	67.59	31.97	85.72
Checkpoint at epoch 55	43.36	33.47	59.63	62.48	23.27	74.73	67.75	32.11	85.82
Checkpoint at epoch 60	43.75	33.38	59.41	62.35	23.31	74.58	67.72	32.60	85.87
Checkpoint at epoch 65	44.68	32.42	61.17	62.19	23.47	74.42	67.66	32.86	85.88
Checkpoint at epoch 70	45.67	31.84	62.83	61.82	23.98	73.70	67.61	33.14	85.84
Checkpoint at epoch 75	45.75	32.68	62.63	61.95	24.09	73.58	67.45	33.44	85.86
Checkpoint at epoch 80	45.00	33.51	61.41	62.17	24.38	74.13	67.27	33.64	85.79
Checkpoint at epoch 85	41.68	39.02	57.90	62.22	24.42	74.54	66.98	34.08	85.60
Checkpoint at epoch 90	39.08	44.51	55.08	61.96	24.35	74.44	66.90	34.41	85.41
Checkpoint at epoch 95	38.79	45.10	55.29	61.60	24.62	74.17	67.01	34.52	85.39
Checkpoint at epoch 100	40.02	42.83	57.45	61.50	25.11	73.98	67.12	34.61	85.48
Checkpoint ensemble	45.58	16.96	63.24	62.25	19.99	75.30	67.74	29.14	85.87

Table 22: Evaluating the checkpoints during fine-tuning and the checkpoint ensemble constructed after fine-tuning on the target test dataset. All numbers are percentages.

F.11 Training with unsupervised domain adaptation

In this section, we study whether incorporating Unsupervised Domain Adaptation (UDA) techniques into training could improve the selective prediction performance. UDA techniques are mainly

proposed to adapt the representation learned on the labeled source domain data to the target domain with unlabeled data from the target domain (Liu et al., 2022). We can easily incorporate those UDA techniques into SR (Algorithm 2), DE (Algorithm 3), and the proposed ASPEST (Algorithm 1) by adding unsupervised training losses into the training objectives.

We consider the method `DE with UDA` and the method `ASPEST with UDA`. The algorithm for DE with UDA is presented in Algorithm 4 and the algorithm for ASPEST with UDA is presented in Algorithm 5. We consider UDA techniques based on representation matching where the goal is to minimize the distance between the distribution of the representation on $\mathcal{D}^{\text{tr}}$ and that on U_X. Suppose the model f is a composition of a prediction function h and a representation function ϕ (i.e., $f(x) = h(\phi(x))$). Then $L_{UDA}(\mathcal{D}^{\text{tr}}, U_X; \theta) = d(p^{\phi}_{\mathcal{D}^{\text{tr}}}, p^{\phi}_{U_X})$, which is a representation matching loss. We consider the representation matching losses from the state-of-the-art UDA methods DANN (Ganin et al., 2016) and CDAN (Long et al., 2018).

We evaluate two instantiations of Algorithm 4 – `DE with DANN` and `DE with CDAN`, and two instantiations of Algorithm 5 – `ASPEST with DANN` and `ASPEST with CDAN`. The values of the hyper-parameters are the same as those described in the paper except that we set $n_s = 20$. For DANN and CDAN, we set the hyper-parameter between the source classifier and the domain discriminator to be 0.1. The results are shown in Table 23 (MNIST→SVHN), Table 24 (CIFAR-10→CINIC-10), Table 25 (FMoW), Table 26 (Amazon Review), Table 27 (DomainNet R→C), Table 28 (DomainNet R→P), Table 29 (DomainNet R→S) and Table 30 (Otto).

From the results, we can see that ASPEST outperforms (or on par with) `DE with DANN` and `DE with CDAN` across different datasets, although ASPEST doesn't use UDA techniques. We further show that by combining ASPEST with DANN or CDAN, it can achieve even better performance. We find some cases where combining ASPEST with DANN or CDAN leads to much worse results. For example, on MNIST→SVHN, when the labeling budget is very small (i.e., 100), combining ASPEST with DANN or CDAN leads to much worse results. It might be because in these cases, DANN or CDAN fails to align the representations between the source and target domains.

Dataset	MNIST→SVHN								
Metric	$cov\|acc \geq 90\% \uparrow$			$acc\|cov \geq 90\% \uparrow$			AUC ↑		
Labeling Budget	100	500	1000	100	500	1000	100	500	1000
DE with DANN + Uniform	27.27±1.8	72.78±2.0	87.05±0.5	63.95±1.4	82.99±0.8	88.64±0.2	80.37±0.7	93.25±0.4	96.05±0.1
DE with DANN + Entropy	11.33±8.2	74.04±2.2	91.06±1.4	58.28±2.1	83.64±0.9	90.41±0.5	74.62±1.6	93.45±0.5	96.47±0.2
DE with DANN + Confidence	15.68±6.3	76.34±3.1	93.96±1.2	61.32±3.0	85.02±0.9	91.64±0.4	76.43±3.0	93.85±0.6	96.97±0.3
DE with DANN + Margin	30.64±2.1	83.44±0.9	96.17±0.5	66.79±0.9	87.30±0.4	92.71±0.2	82.14±0.8	95.40±0.3	97.60±0.1
DE with DANN + Avg-KLD	22.30±3.0	78.13±2.1	93.42±1.0	63.22±2.0	85.40±0.8	91.47±0.5	78.88±1.6	94.25±0.5	97.02±0.2
DE with DANN + CLUE	16.42±13.6	72.27±2.8	86.71±0.4	61.79±2.7	82.72±1.1	88.46±0.2	77.47±3.4	93.33±0.5	96.21±0.0
DE with DANN + BADGE	25.41±10.9	78.83±1.2	90.94±1.1	63.93±4.4	85.27±0.5	90.45±0.5	79.82±4.1	94.58±0.3	96.89±0.1
DE with CDAN + Uniform	28.10±4.8	73.15±0.7	87.50±0.6	63.95±2.7	83.10±0.3	88.86±0.3	80.28±2.2	93.44±0.1	96.13±0.2
DE with CDAN + Entropy	6.94±9.8	74.38±1.5	90.77±1.3	59.90±2.3	84.14±0.4	90.32±0.6	76.04±2.0	93.48±0.3	96.38±0.2
DE with CDAN + Confidence	13.47±10.2	75.15±2.8	92.77±0.7	60.98±2.0	84.62±0.9	91.23±0.3	76.19±2.8	93.62±0.6	96.63±0.1
DE with CDAN + Margin	22.44±3.3	81.84±2.5	96.07±0.2	62.89±3.8	86.71±1.0	92.64±0.0	78.69±2.6	94.89±0.5	97.57±0.0
DE with CDAN + Avg-KLD	20.23±4.1	80.62±1.7	93.13±2.5	62.23±2.7	86.34±0.6	91.30±1.0	77.68±2.5	94.81±0.4	96.97±0.4
DE with CDAN + CLUE	7.47±6.4	72.61±2.9	87.22±0.2	57.82±2.9	82.50±1.3	88.62±0.1	73.33±2.3	93.38±0.7	96.31±0.0
DE with CDAN + BADGE	26.88±3.5	79.21±0.1	92.50±0.7	65.69±1.7	85.32±0.1	91.18±0.4	81.10±1.3	94.73±0.1	97.17±0.2
ASPEST	**52.10**±4.0	89.22±0.9	98.70±0.4	**76.10**±1.5	89.62±0.4	93.92±0.3	**88.84**±1.0	96.62±0.2	98.06±0.1
ASPEST with DANN	37.90±2.4	**91.61**±0.6	99.39±0.4	69.45±1.7	**90.70**±0.3	94.42±0.4	84.55±1.0	**97.03**±0.1	98.23±0.1
ASPEST with CDAN	30.97±11.7	91.39±0.6	**99.50**±0.3	67.58±3.2	90.60±0.3	**94.46**±0.2	82.20±3.3	96.95±0.1	**98.26**±0.1

Table 23: Results of evaluating DE with UDA and ASPEST with UDA on MNIST→SVHN. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Algorithm 4 DE with Unsupervised Domain Adaptation

Input: A training dataset $\mathcal{D}^{\text{tr}}$, An unlabeled test dataset U_X, the number of rounds T, the total labeling budget M, a source-trained model $\bar{f}$, an acquisition function $a(B, f, g)$, the number of models in the ensemble N, the number of initial training epochs n_s, and a hyper-parameter λ.

Let $f_0^j = \bar{f}$ for $j = 1, \ldots, N$.

Fine-tune each model f_0^j in the ensemble via SGD for n_s training epochs independently using the following training objective with different randomness:

$$\min_{\theta^j} \quad \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) + L_{UDA}(\mathcal{D}^{\text{tr}}, U_X; \theta^j) \tag{37}$$

where L_{UDA} is a loss function for unsupervised domain adaptation.

Let $B_0 = \emptyset$.

Let $g_t(\mathbf{x}) = \max_{k\in\mathcal{Y}} f_t(\mathbf{x} \mid k)$.

for $t = 1, \cdots, T$ **do**

Select a batch B_t with a size of $m = [\frac{M}{T}]$ from U_X for labeling via:

$$B_t = \arg \max_{B\subset U_X \setminus (\cup_{l=0}^{t-1} B_l), |B|=m} a(B, f_{t-1}, g_{t-1}) \tag{38}$$

Use an oracle to assign ground-truth labels to the examples in B_t to get $\tilde{B}_t$.

Fine-tune each model f_{t-1}^j in the ensemble via SGD independently using the following training objective with different randomness:

$$\min_{\theta^j} \quad \mathbb{E}_{(\mathbf{x},y)\in\cup_{l=1}^{t}\tilde{B}_l} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) + \lambda \cdot \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) + L_{UDA}(\mathcal{D}^{\text{tr}}, U_X; \theta^j) \tag{39}$$

where θ^j is the model parameters of f_{t-1}^j.

Let $f_t^j = f_{t-1}^j$.

end for

Output: The classifier $f = f_T$ and the selection scoring function $g = \max_{k\in\mathcal{Y}} f(\mathbf{x} \mid k)$.

Dataset	CIFAR-10→CINIC-10								
Metric	$cov\|acc \geq 90\% \uparrow$			$acc\|cov \geq 90\% \uparrow$			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
DE with DANN + Uniform	58.85±0.3	59.39±0.2	60.04±0.1	77.06±0.2	77.33±0.2	77.84±0.1	90.40±0.1	90.60±0.1	90.73±0.1
DE with DANN + Entropy	59.42±0.4	60.86±0.3	64.52±0.3	78.14±0.2	79.20±0.1	81.31±0.1	90.72±0.0	91.06±0.1	92.02±0.0
DE with DANN + Confidence	59.44±0.6	61.08±0.3	65.12±0.2	78.19±0.1	79.38±0.0	81.29±0.1	90.73±0.1	91.26±0.1	92.06±0.0
DE with DANN + Margin	59.81±0.3	62.26±0.4	65.58±0.4	78.15±0.0	79.25±0.1	81.05±0.1	90.76±0.1	91.30±0.1	92.11±0.0
DE with DANN + Avg-KLD	60.50±0.5	62.04±0.1	65.08±0.2	78.32±0.1	79.31±0.1	81.07±0.0	90.89±0.1	91.34±0.0	92.11±0.0
DE with DANN + CLUE	60.20±0.5	61.69±0.2	64.08±0.2	77.84±0.2	78.35±0.2	79.38±0.1	90.73±0.2	91.07±0.1	91.63±0.0
DE with DANN + BADGE	60.18±0.4	62.15±0.2	65.31±0.6	77.70±0.1	78.54±0.1	79.81±0.2	90.72±0.1	91.19±0.1	91.86±0.1
DE with CDAN + Uniform	58.72±0.2	59.49±0.5	60.28±0.2	77.16±0.0	77.52±0.1	77.90±0.1	90.45±0.1	90.65±0.0	90.78±0.1
DE with CDAN + Entropy	58.73±0.4	60.82±0.5	64.45±0.2	77.95±0.1	79.20±0.1	81.04±0.1	90.57±0.1	91.10±0.1	91.86±0.1
DE with CDAN + Confidence	59.10±0.6	61.03±0.6	64.60±0.2	77.92±0.0	79.26±0.2	81.07±0.0	90.59±0.0	91.10±0.2	91.96±0.0
DE with CDAN + Margin	59.88±0.5	61.57±0.9	64.82±0.4	78.09±0.3	79.02±0.2	80.82±0.1	90.73±0.1	91.17±0.2	91.98±0.1
DE with CDAN + Avg-KLD	60.51±0.1	61.71±0.5	65.03±0.3	78.20±0.2	79.29±0.2	81.15±0.1	90.85±0.0	91.19±0.1	92.07±0.1
DE with CDAN + CLUE	60.12±0.5	61.77±0.3	64.06±0.2	77.88±0.1	78.38±0.2	79.42±0.2	90.73±0.1	91.08±0.1	91.64±0.0
DE with CDAN + BADGE	60.28±0.7	61.84±0.2	65.29±0.3	77.68±0.2	78.53±0.1	79.84±0.2	90.73±0.1	91.17±0.0	91.95±0.1
ASPEST	60.38±0.3	63.34±0.2	66.81±0.3	78.23±0.1	79.49±0.1	81.25±0.1	90.95±0.0	91.60±0.0	92.33±0.1
ASPEST with DANN	**61.69**±0.2	**63.58**±0.4	**66.81**±0.4	**78.68**±0.1	**79.68**±0.1	81.42±0.1	**91.16**±0.1	**91.66**±0.1	92.37±0.1
ASPEST with CDAN	61.00±0.2	62.80±0.4	66.78±0.1	78.56±0.1	79.54±0.1	**81.49**±0.0	91.13±0.0	91.57±0.1	**92.41**±0.0

Table 24: Results of evaluating DE with UDA and ASPEST with UDA on CIFAR-10→CINIC-10. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Algorithm 5 ASPEST with Unsupervised Domain Adaptation

Input: A training set $\mathcal{D}^{tr}$, a unlabeled test set U_X, the number of rounds T, the labeling budget M, the number of models N, the number of initial training epochs n_s, a checkpoint epoch c_e, a threshold η, a sub-sampling fraction p, and a hyper-parameter λ.

Let $f_0^j = \bar{f}$ for $j = 1, \ldots, N$.

Set $N_e = 0$ and $P = \mathbf{0}_{n \times K}$.

Fine-tune each f_0^j for n_s training epochs using the following training objective:

$$\min_{\theta^j} \quad \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) + L_{UDA}(\mathcal{D}^{\text{tr}}, U_X; \theta^j), \tag{40}$$

where L_{UDA} is a loss function for unsupervised domain adaptation. During fine-tuning, update P and N_e using Eq. (8) every c_e training epochs.

for $t = 1, \cdots, T$ **do**

Select a batch B_t from U_X for labeling using the sample selection objective (10).

Use an oracle to assign ground-truth labels to the examples in B_t to get $\tilde{B}_t$.

Set $N_e = 0$ and $P = \mathbf{0}_{n \times K}$.

Fine-tune each f_{t-1}^j using the following training objective:

$$\min_{\theta^j} \quad \mathbb{E}_{(\mathbf{x},y)\in\cup_{l=1}^{t}\tilde{B}_l} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) + \lambda \cdot \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) + L_{UDA}(\mathcal{D}^{\text{tr}}, U_X; \theta^j), \tag{41}$$

During fine-tuning, update P and N_e using Eq (8) every c_e training epochs.

Let $f_t^j = f_{t-1}^j$.

Construct the pseudo-labeled set R via Eq (12) and create R_{sub} by randomly sampling up to $\lfloor p \cdot n \rfloor$ data points from R.

Train each f_t^j further via SGD using the objective (13) and update P and N_e using Eq (8) every c_e training epochs.

end for

Output: The classifier $f(\mathbf{x}_i) = \arg\max_{k\in\mathcal{Y}} P_{i,k}$ and the selection scoring function $g(\mathbf{x}_i) = \max_{k\in\mathcal{Y}} P_{i,k}$.

Dataset	FMoW								
Metric	$cov\|acc \geq 70\% \uparrow$			$acc\|cov \geq 70\% \uparrow$			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
DE with DANN + Uniform	46.11±0.6	51.77±0.3	62.76±0.5	57.62±0.3	60.67±0.4	66.21±0.2	70.17±0.3	72.46±0.3	76.83±0.2
DE with DANN + Entropy	44.36±0.7	48.19±0.3	59.52±0.8	56.78±0.1	59.51±0.0	65.75±0.3	69.09±0.2	71.02±0.2	75.15±0.3
DE with DANN + Confidence	44.46±0.5	49.32±0.1	61.47±0.3	57.04±0.3	60.51±0.3	66.61±0.1	69.14±0.1	71.50±0.1	75.70±0.1
DE with DANN + Margin	48.09±0.4	54.35±0.5	70.11±0.4	59.07±0.2	62.79±0.2	70.02±0.1	70.76±0.1	73.29±0.2	78.25±0.1
DE with DANN + Avg-KLD	48.42±0.1	55.95±0.2	68.73±1.1	59.06±0.2	63.44±0.2	69.41±0.5	70.84±0.1	73.83±0.1	77.91±0.4
DE with DANN + CLUE	44.14±0.6	46.15±0.2	49.02±0.5	56.01±0.3	56.89±0.2	58.66±0.3	69.11±0.2	70.16±0.2	71.46±0.2
DE with DANN + BADGE	48.57±0.5	54.47±0.5	67.69±0.9	58.61±0.2	61.67±0.0	68.71±0.5	71.17±0.2	73.64±0.1	78.65±0.3
DE with CDAN + Uniform	46.08±0.7	51.92±0.8	62.87±0.2	57.45±0.1	60.73±0.4	66.19±0.2	69.93±0.3	72.57±0.4	76.87±0.1
DE with CDAN + Entropy	44.42±0.3	49.32±0.1	60.11±0.3	56.83±0.1	60.04±0.2	65.95±0.2	69.18±0.2	71.34±0.3	75.44±0.3
DE with CDAN + Confidence	44.75±0.1	49.34±0.1	62.80±1.0	57.09±0.1	60.50±0.2	66.94±0.4	69.27±0.1	71.60±0.2	76.14±0.3
DE with CDAN + Margin	47.48±0.7	54.48±0.7	70.25±0.9	58.98±0.4	62.98±0.3	70.10±0.4	70.55±0.3	73.46±0.2	78.39±0.3
DE with CDAN + Avg-KLD	48.43±0.2	54.37±0.4	68.93±0.6	59.36±0.2	62.71±0.2	69.54±0.2	71.12±0.2	73.35±0.2	77.97±0.2
DE with CDAN + CLUE	44.09±0.3	46.11±0.5	48.90±0.1	55.78±0.3	56.98±0.2	58.46±0.2	69.03±0.1	70.02±0.2	71.31±0.1
DE with CDAN + BADGE	47.93±0.2	54.61±0.2	67.01±0.5	58.16±0.1	61.81±0.1	68.36±0.2	70.91±0.2	73.63±0.1	78.52±0.2
ASPEST	**53.05**±0.4	**59.86**±0.4	**76.52**±0.6	61.18±0.2	**65.18**±0.2	**72.86**±0.3	71.12±0.2	**74.25**±0.2	**79.93**±0.1
ASPEST with DANN	51.02±0.9	58.63±1.1	72.97±0.9	61.10±0.5	64.98±0.4	71.21±0.4	71.03±0.3	73.79±0.4	77.84±0.3
ASPEST with CDAN	51.40±0.6	58.21±0.6	73.94±0.6	**61.38**±0.2	65.04±0.2	71.63±0.2	**71.17**±0.1	73.59±0.1	78.04±0.2

Table 25: Results of evaluating DE with UDA and ASPEST with UDA on FMoW. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	Amazon Review								
Metric	$cov\|acc \geq 80\% \uparrow$			$acc\|cov \geq 80\% \uparrow$			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
DE with DANN + Uniform	38.55±3.3	37.25±1.8	39.21±1.9	69.06±0.6	68.94±0.1	69.41±0.2	77.52±0.7	77.03±0.4	77.70±0.2
DE with DANN + Entropy	38.22±2.3	41.85±0.8	41.57±1.3	69.48±0.3	**70.71**±0.3	71.55±0.2	77.49±0.5	78.39±0.2	78.58±0.1
DE with DANN + Confidence	38.01±1.0	38.36±2.5	38.89±1.3	69.45±0.1	70.16±0.3	71.44±0.2	77.54±0.2	77.58±0.5	78.48±0.3
DE with DANN + Margin	36.82±1.3	36.89±1.3	41.98±1.5	69.35±0.3	69.63±0.3	71.27±0.2	77.30±0.3	77.23±0.3	78.34±0.3
DE with DANN + Avg-KLD	37.15±2.9	38.21±1.3	42.46±1.4	69.38±0.4	69.79±0.2	71.21±0.2	77.25±0.6	77.72±0.3	78.68±0.3
DE with DANN + CLUE	**40.23**±4.0	34.71±1.8	31.38±0.9	68.95±0.7	68.07±0.2	67.44±0.3	77.62±1.0	76.27±0.6	75.60±0.2
DE with DANN + BADGE	37.51±1.8	37.00±0.9	41.62±2.3	68.98±0.4	69.27±0.1	70.20±0.4	77.20±0.4	77.21±0.1	78.31±0.5
DE with CDAN + Uniform	37.81±0.3	37.83±2.7	39.52±0.8	68.93±0.1	69.16±0.7	69.50±0.3	77.16±0.1	77.30±0.7	77.74±0.3
DE with CDAN + Entropy	37.99±0.8	37.68±1.1	42.55±0.9	**69.54**±0.3	70.01±0.2	71.52±0.2	77.52±0.2	77.61±0.1	78.63±0.1
DE with CDAN + Confidence	35.76±0.9	38.69±2.8	41.43±2.1	69.24±0.0	70.45±0.4	71.50±0.4	77.08±0.2	77.82±0.4	78.47±0.3
DE with CDAN + Margin	37.68±2.9	37.43±1.0	42.18±1.3	69.50±0.3	69.80±0.4	71.29±0.0	77.50±0.5	77.31±0.3	78.46±0.3
DE with CDAN + Avg-KLD	37.85±1.6	40.71±0.9	44.35±0.9	69.41±0.3	70.29±0.1	71.28±0.2	77.28±0.5	78.11±0.2	78.86±0.2
DE with CDAN + CLUE	34.85±2.7	34.03±1.3	30.70±0.4	68.70±0.3	67.84±0.1	67.12±0.3	76.95±0.7	76.23±0.4	75.36±0.4
DE with CDAN + BADGE	39.47±0.2	39.29±1.1	41.64±0.9	69.33±0.0	69.34±0.2	70.58±0.2	77.52±0.2	77.49±0.2	78.24±0.3
ASPEST	38.44±0.7	40.96±0.8	45.77±0.1	69.31±0.3	70.17±0.2	**71.60**±0.2	77.69±0.1	78.35±0.2	**79.51**±0.2
ASPEST with DANN	40.22±0.5	41.99±1.4	**45.84**±0.1	69.42±0.1	70.30±0.1	71.58±0.2	**78.00**±0.1	78.34±0.3	79.43±0.1
ASPEST with CDAN	40.02±0.5	**42.46**±0.6	44.95±0.4	69.50±0.1	70.37±0.2	71.42±0.0	77.80±0.1	**78.57**±0.1	79.25±0.0

Table 26: Results of evaluating DE with UDA and ASPEST with UDA on Amazon Review. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	DomainNet R→C (easy)								
Metric	$cov\|acc \geq 80\% \uparrow$			$acc\|cov \geq 80\% \uparrow$			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
DE with DANN + Uniform	33.53±0.5	36.28±0.3	40.13±1.0	50.57±0.5	52.19±0.1	55.15±0.1	69.34±0.3	70.98±0.2	73.50±0.3
DE with DANN + Entropy	28.66±1.0	34.47±0.1	42.77±0.7	48.13±0.6	52.70±0.3	59.01±0.2	66.60±0.5	70.64±0.1	75.45±0.2
DE with DANN + Confidence	29.92±0.4	35.29±1.0	43.33±0.4	48.61±0.1	53.36±0.5	59.72±0.3	67.23±0.2	70.92±0.5	75.89±0.3
DE with DANN + Margin	35.19±0.3	39.63±0.2	46.51±0.5	52.29±0.3	55.60±0.2	60.97±0.4	70.70±0.1	73.41±0.1	77.24±0.3
DE with DANN + Avg-KLD	36.02±0.6	39.67±0.5	47.20±0.8	53.00±0.3	55.75±0.3	61.22±0.3	71.19±0.3	73.51±0.2	77.46±0.2
DE with DANN + CLUE	32.26±1.5	35.09±0.4	35.66±0.3	50.21±0.0	50.90±0.1	51.50±0.1	69.17±0.2	70.20±0.2	70.82±0.1
DE with DANN + BADGE	35.27±0.5	38.88±0.3	45.97±0.7	52.15±0.3	54.89±0.1	60.03±0.3	70.65±0.1	72.95±0.1	76.87±0.1
DE with CDAN + Uniform	33.49±0.6	36.01±0.7	39.93±0.2	50.46±0.2	51.89±0.1	55.23±0.2	69.32±0.3	70.86±0.3	73.55±0.2
DE with CDAN + Entropy	29.50±0.5	33.86±0.3	42.24±0.5	48.01±0.1	52.52±0.3	58.96±0.2	66.82±0.2	70.28±0.1	75.33±0.1
DE with CDAN + Confidence	29.21±1.0	34.92±0.6	43.36±0.4	48.48±0.4	52.85±0.4	59.88±0.4	66.82±0.5	70.61±0.4	75.93±0.3
DE with CDAN + Margin	35.87±0.7	38.37±0.4	46.42±0.6	52.58±0.1	55.28±0.2	61.20±0.2	70.95±0.2	72.95±0.2	77.26±0.1
DE with CDAN + Avg-KLD	36.21±0.6	40.08±0.3	47.62±0.4	52.95±0.3	55.93±0.1	61.56±0.2	71.29±0.3	73.60±0.1	77.58±0.2
DE with CDAN + CLUE	31.74±2.1	35.11±0.2	35.87±0.5	49.99±0.2	51.39±0.2	51.43±0.2	69.04±0.3	70.35±0.0	70.82±0.3
DE with CDAN + BADGE	34.74±0.5	38.68±0.7	45.87±1.0	51.80±0.3	54.75±0.2	60.22±0.1	70.38±0.1	72.90±0.2	76.85±0.2
ASPEST	37.38±0.1	39.98±0.3	48.29±1.0	54.56±0.3	56.95±0.1	62.69±0.2	71.61±0.2	73.27±0.2	77.40±0.4
ASPEST with DANN	**37.41**±0.8	42.45±1.0	49.74±0.6	**55.60**±0.1	58.29±0.2	63.64±0.2	71.88±0.2	74.18±0.4	78.09±0.0
ASPEST with CDAN	36.60±1.2	**42.96**±0.6	**50.86**±0.2	55.55±0.2	**58.71**±0.2	**63.85**±0.2	**71.99**±0.2	**74.60**±0.2	**78.45**±0.3

Table 27: Results of evaluating DE with UDA and ASPEST with UDA on DomainNet R→C. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	DomainNet R→P (medium)								
Metric	$cov\|acc \geq 70\% \uparrow$			$acc\|cov \geq 70\% \uparrow$			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
DE with DANN + Uniform	26.98±0.1	28.34±0.5	30.63±0.2	41.96±0.2	42.89±0.2	44.73±0.1	57.04±0.1	58.10±0.2	59.87±0.1
DE with DANN + Entropy	24.75±0.4	27.02±0.5	30.10±0.2	40.29±0.4	42.34±0.2	45.78±0.2	55.19±0.3	57.12±0.3	60.21±0.1
DE with DANN + Confidence	22.41±0.9	27.03±0.6	31.70±0.6	39.05±0.5	42.61±0.2	46.60±0.2	53.66±0.6	57.35±0.3	60.93±0.4
DE with DANN + Margin	29.16±0.1	30.58±0.3	33.64±0.6	43.78±0.2	45.17±0.2	47.69±0.4	58.76±0.1	59.94±0.0	62.19±0.4
DE with DANN + Avg-KLD	29.52±0.1	31.17±0.4	34.09±0.3	43.84±0.3	45.33±0.2	48.18±0.2	58.89±0.2	60.25±0.2	62.54±0.2
DE with DANN + CLUE	27.48±0.5	27.83±0.2	28.39±0.5	42.05±0.3	42.34±0.2	42.65±0.1	57.32±0.3	57.64±0.2	57.99±0.2
DE with DANN + BADGE	28.92±0.1	30.36±0.2	33.86±0.3	43.38±0.1	44.85±0.1	47.64±0.3	58.38±0.0	59.82±0.1	62.26±0.2
DE with CDAN + Uniform	26.96±0.4	28.33±0.2	29.98±0.4	41.77±0.3	42.85±0.2	44.23±0.4	56.86±0.4	58.01±0.0	59.42±0.4
DE with CDAN + Entropy	24.91±0.4	26.30±0.9	30.33±0.4	40.34±0.3	42.07±0.6	45.79±0.2	55.38±0.4	56.70±0.8	60.23±0.2
DE with CDAN + Confidence	24.58±0.7	27.11±0.5	31.07±0.5	40.32±0.2	42.64±0.3	46.25±0.3	55.14±0.3	57.40±0.3	60.63±0.3
DE with CDAN + Margin	28.33±0.1	30.17±0.3	33.54±0.4	43.44±0.4	44.77±0.1	47.56±0.2	58.31±0.2	59.65±0.1	62.17±0.2
DE with CDAN + Avg-KLD	28.69±0.2	30.99±0.9	34.30±0.2	43.64±0.2	45.34±0.2	48.22±0.1	58.60±0.1	60.15±0.4	62.67±0.1
DE with CDAN + CLUE	27.52±0.6	27.96±0.2	28.18±0.5	42.02±0.2	42.44±0.1	42.67±0.2	57.21±0.3	57.70±0.1	58.04±0.3
DE with CDAN + BADGE	28.79±0.1	30.28±0.1	33.77±0.4	43.45±0.0	44.73±0.3	47.84±0.2	58.47±0.1	59.64±0.2	62.37±0.2
ASPEST	29.69±0.1	32.50±0.3	35.46±0.6	44.96±0.1	46.77±0.2	49.42±0.1	58.74±0.0	60.36±0.0	62.84±0.2
ASPEST with DANN	**31.75**±0.4	**33.58**±0.3	36.96±0.2	**46.16**±0.1	47.64±0.2	**50.37**±0.3	**59.63**±0.2	61.06±0.1	**63.75**±0.1
ASPEST with CDAN	30.39±0.4	33.57±0.3	**37.53**±0.7	45.90±0.1	**47.71**±0.2	50.31±0.2	59.13±0.3	**61.17**±0.2	63.69±0.3

Table 28: Results of evaluating DE with UDA and ASPEST with UDA on DomainNet R→P. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	DomainNet R→S (hard)								
Metric	$cov\|acc \geq 70\% \uparrow$			$acc\|cov \geq 70\% \uparrow$			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
DE with DANN + Uniform	17.55±0.4	19.82±0.3	23.57±0.4	32.61±0.5	34.56±0.3	37.73±0.2	47.60±0.5	49.92±0.4	53.52±0.1
DE with DANN + Entropy	10.77±0.8	15.38±0.5	20.11±0.5	27.78±0.7	31.09±0.2	36.39±0.3	41.69±0.7	45.62±0.3	51.05±0.4
DE with DANN + Confidence	10.64±1.2	15.22±0.4	20.25±0.5	28.09±1.0	31.76±0.3	36.86±0.8	41.94±1.3	46.19±0.3	51.48±0.7
DE with DANN + Margin	17.90±0.7	20.44±0.6	25.52±0.4	33.61±0.1	35.79±0.5	40.29±0.3	48.67±0.1	51.03±0.6	55.64±0.4
DE with DANN + Avg-KLD	18.02±1.0	21.22±0.2	25.46±0.2	34.00±0.2	36.51±0.2	40.72±0.2	49.05±0.2	51.79±0.2	55.95±0.2
DE with DANN + CLUE	15.77±0.3	18.14±0.7	19.49±0.4	32.10±0.1	33.42±0.3	34.50±0.3	47.18±0.2	48.63±0.3	50.03±0.3
DE with DANN + BADGE	16.84±0.9	20.88±0.3	25.11±0.3	33.97±0.1	36.20±0.2	40.01±0.3	48.87±0.2	51.46±0.2	55.33±0.2
DE with CDAN + Uniform	17.33±0.5	19.79±0.1	22.99±0.5	32.47±0.5	34.59±0.3	37.88±0.2	47.49±0.5	50.02±0.2	53.51±0.3
DE with CDAN + Entropy	12.48±0.8	15.19±0.8	20.23±0.0	28.83±0.1	32.41±0.4	36.57±0.1	42.93±0.5	47.00±0.3	51.24±0.2
DE with CDAN + Confidence	11.23±0.6	13.93±0.1	18.45±1.3	28.67±0.3	31.35±0.4	35.56±0.8	42.87±0.5	45.40±0.7	49.80±1.0
DE with CDAN + Margin	18.06±0.7	20.39±0.3	25.05±0.3	33.98±0.2	35.76±0.2	40.11±0.1	49.15±0.1	50.92±0.1	55.27±0.1
DE with CDAN + Avg-KLD	18.63±1.0	20.80±0.3	25.49±0.9	34.19±0.4	36.41±0.2	40.53±0.5	49.45±0.5	51.58±0.1	55.74±0.5
DE with CDAN + CLUE	16.51±0.3	18.82±0.1	19.47±0.1	32.23±0.2	33.83±0.4	34.72±0.3	47.40±0.2	49.11±0.2	49.98±0.3
DE with CDAN + BADGE	17.52±0.8	21.48±0.5	25.35±0.4	33.53±0.5	36.19±0.4	40.31±0.3	48.67±0.5	51.65±0.3	55.62±0.3
ASPEST	17.86±0.4	20.42±0.4	25.87±0.4	35.17±0.1	37.28±0.3	41.46±0.2	49.62±0.1	51.61±0.4	55.90±0.2
ASPEST with DANN	16.35±1.2	**23.18**±0.4	28.00±0.1	36.56±0.2	**39.40**±0.4	42.94±0.1	50.58±0.4	**53.73**±0.3	57.25±0.1
ASPEST with CDAN	**18.81**±1.1	22.95±0.8	**28.17**±0.2	**36.85**±0.3	39.10±0.2	**43.25**±0.3	**51.14**±0.3	53.47±0.2	**57.26**±0.2

Table 29: Results of evaluating DE with UDA and ASPEST with UDA on DomainNet R→S. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	Otto								
Metric	$cov\|acc \geq 80\% \uparrow$			$acc\|cov \geq 80\% \uparrow$			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
DE with DANN + Uniform	70.35±0.5	72.42±0.4	75.63±0.7	76.12±0.3	77.04±0.1	78.25±0.1	86.67±0.1	87.16±0.1	88.09±0.1
DE with DANN + Entropy	75.27±0.3	81.25±0.1	92.23±0.3	78.14±0.1	80.45±0.0	83.73±0.1	87.73±0.1	88.91±0.0	90.90±0.1
DE with DANN + Confidence	74.66±0.3	81.62±0.1	92.57±0.6	78.05±0.2	80.50±0.0	83.67±0.2	87.51±0.1	89.06±0.1	90.94±0.1
DE with DANN + Margin	75.47±0.4	82.56±0.7	91.86±0.9	78.26±0.1	80.79±0.2	83.61±0.3	87.87±0.1	89.08±0.0	90.88±0.1
DE with DANN + Avg-KLD	76.02±0.6	81.78±0.4	91.82±0.3	78.53±0.0	80.70±0.1	83.88±0.0	87.99±0.0	89.17±0.0	90.90±0.1
DE with DANN + CLUE	69.68±0.4	68.07±0.3	62.70±0.6	75.81±0.3	75.44±0.0	73.49±0.3	86.68±0.2	86.31±0.1	84.89±0.2
DE with DANN + BADGE	74.69±0.5	79.04±0.6	87.63±0.4	77.97±0.1	79.57±0.3	82.99±0.1	87.82±0.1	88.92±0.1	90.67±0.1
DE with CDAN + Uniform	70.25±0.9	72.43±0.4	75.21±0.7	76.09±0.3	76.94±0.1	78.13±0.1	86.56±0.3	87.14±0.2	87.90±0.1
DE with CDAN + Entropy	74.73±0.6	81.60±0.8	92.58±0.2	77.97±0.2	80.59±0.3	83.81±0.2	87.47±0.1	88.93±0.1	90.84±0.1
DE with CDAN + Confidence	74.88±0.6	81.30±0.8	92.53±0.9	78.06±0.2	80.51±0.3	83.85±0.3	87.43±0.2	88.99±0.1	90.95±0.1
DE with CDAN + Margin	76.68±1.0	81.57±0.4	92.20±0.5	78.74±0.5	80.62±0.2	84.01±0.2	88.08±0.2	88.85±0.2	91.09±0.0
DE with CDAN + Avg-KLD	75.88±0.4	81.82±0.8	91.43±1.1	78.45±0.1	80.72±0.3	83.72±0.3	87.92±0.2	89.12±0.2	90.91±0.2
DE with CDAN + CLUE	69.86±0.5	67.79±0.2	63.46±0.9	76.09±0.2	75.42±0.3	73.66±0.3	86.81±0.1	86.25±0.1	85.00±0.1
DE with CDAN + BADGE	74.68±0.4	79.46±0.3	87.57±0.4	77.89±0.1	79.78±0.1	82.85±0.1	87.78±0.1	88.90±0.1	90.72±0.1
ASPEST	77.85±0.2	**84.20**±0.6	94.26±0.6	79.28±0.1	**81.40**±0.1	84.62±0.1	88.28±0.1	**89.61**±0.1	**91.49**±0.0
ASPEST with DANN	**78.14**±0.4	83.33±0.5	93.61±0.0	**79.33**±0.1	81.23±0.1	84.21±0.1	**88.36**±0.2	89.32±0.1	91.26±0.0
ASPEST with CDAN	77.75±0.3	83.68±0.5	**94.44**±0.3	79.27±0.0	81.30±0.2	**84.76**±0.1	88.35±0.1	89.59±0.0	91.41±0.0

Table 30: Results of evaluating DE with UDA and ASPEST with UDA on Otto. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.