Selective Prediction via Training Dynamics

Stephan Rabanser[1*] **Anvith Thudi**[1] **Kimia Hamidieh**[2] **Adam Dziedzic**[3] **Israfil Bahceci**[4] **Akram Bin Sediq**[4] **Hamza Sokun**[4] **Nicolas Papernot**[1]

[1]*University of Toronto & Vector Institute* [2]*Massachusetts Institute of Technology* [3]*CISPA Helmholtz Center for Information Security* [4]*Ericsson*

Abstract

Selective prediction aims to reject inputs a model is likely to misclassify, balancing input coverage (how many points are accepted) with utility (performance on accepted inputs). Existing methods often modify model architectures or objectives, limiting practical use and introducing unwanted interactions with existing losses. In contrast, we show that state-of-the-art performance can be achieved by analyzing a model's discretized training dynamics. Our framework monitors the instability of intermediate checkpoint predictions relative to the final model and rejects inputs with excessive late-stage disagreement. The approach is domain-agnostic, requires no train-time changes, and can be combined with existing methods. Experiments across image, regression, and time series tasks show that our method outperforms prior state-of-the-art utility–coverage trade-offs.

1. Introduction

Machine learning (ML) systems are increasingly deployed in high-stakes domains—such as healthcare [9, 36], self-driving [19], and law [46]—where prediction errors have severe consequences. To mitigate such risks, *Selective Prediction* (SP) augments models with a gating mechanism that abstains on inputs deemed uncertain [16], aiming to maintain high utility while maximizing coverage.

Existing SP techniques either modify the model architecture [17] or require retraining with custom loss functions [15, 27, 34]. These interventions are often impractical in production environments that impose constraints such as restricted data access, high retraining costs, or fixed pipelines. As a result, current approaches are difficult to deploy at scale. In this work, we show that such modifications are unnecessary. Our method achieves state-of-the-art selective prediction performance across diverse datasets and tasks—while requiring only access to standard training checkpoints. It applies not only to classification, but also to regression and time series prediction tasks, addressing a gap in recent SP literature which has focused almost exclusively on classification.

Our key insight is that iteratively training a deep neural network with stochastic gradient updates implicitly generates a sequence of intermediate models. These training dynamics contain valuable information: past work [1, 25, 30, 45] has linked convergence speed to example difficulty on training points, and we hypothesize that similar signals exist for test-time uncertainty. Specifically, we show that instability in intermediate predictions often correlates with predictive uncertainty, and propose to quantify this via a weighted instability score that emphasizes late-stage disagreement in intermediate models.

Building on this insight, we introduce **S**elective **P**rediction via **T**raining **D**ynamics (`SPTD`). Our approach computes, for each test input, a score that reflects how consistently intermediate models

* Correspondence to: `stephan@cs.toronto.edu`

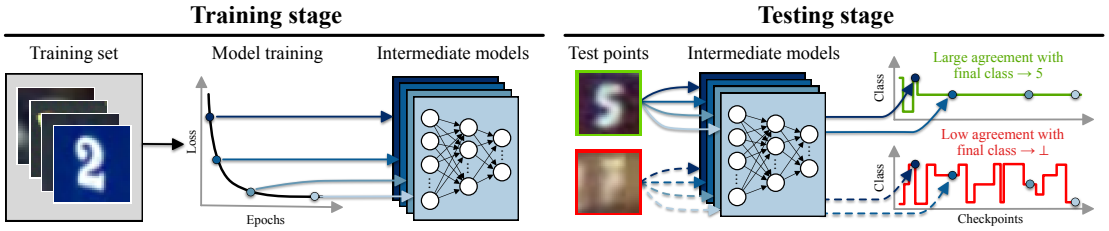


Figure 1: **Overview of our proposed `SPTD` method.** During training, we store intermediate model checkpoints. At test time, we compute how much each test input's predictions agree with the final model. Only inputs with high agreement are accepted.

agree with the final prediction. This score generalizes across domains and does not require retraining. Compared to ensemble-based methods like Deep Ensembles [33], `SPTD` achieves competitive performance at substantially lower training cost while incurring comparable inference-time cost.

2. Background on Selective Prediction

Selective prediction extends supervised learning by allowing models to abstain from uncertain predictions via a rejection option $\perp$ [11]. Inputs $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ are drawn from a covariate space, with labels $y \in \mathcal{Y}$, where $\mathcal{Y} = [C]$ for classification, $\mathbb{R}$ for regression, and $\mathbb{R}^R$ for time series forecasting. A predictive model $f : \mathcal{X} \to \mathcal{Y}$ is paired with a selection function $g : \mathcal{X} \to \mathbb{R}$; given a threshold τ, the model accepts inputs when $g(\boldsymbol{x}) \leq \tau$ and abstains otherwise:

$$(f, g)(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) & g(\boldsymbol{x}) \leq \tau \\ \perp & \text{otherwise.} \end{cases} \tag{1}$$

SP performance is evaluated via a tradeoff between *coverage*—$\text{coverage}(f, g) = \frac{1}{M} \sum_{i=1}^{M} \mathbb{1}[g(\boldsymbol{x}_i) \leq \tau]$, the fraction of inputs accepted—and *selective utility*—$\text{utility}(f, g) = \frac{1}{M_\tau} \sum_{i:g(\boldsymbol{x}_i)\leq\tau} u(f(\boldsymbol{x}_i), y_i)$, the average utility over accepted points, with $M_\tau = \sum_{i=1}^{M} \mathbb{1}[g(\boldsymbol{x}_i) \leq \tau]$. These metrics trade off: improving one often reduces the other. We assess performance over the full utility–coverage spectrum via risk–coverage curves [11], and define task-specific utility functions $u(\cdot, \cdot)$ in Appendix A.

Related Work. Prior work on selective prediction has primarily focused on classification. A popular method is the softmax response (`SR`) baseline [16, 24], which thresholds confidence from the softmax output but suffers from miscalibration [21]. Subsequent methods modify model architectures or loss functions—such as SelectiveNet [17], Deep Gamblers [34], and SAT [27]—to encourage rejection, often under coverage constraints. These typically require retraining or architectural changes, limiting deployment. Uncertainty quantification methods like deep ensembles [33] and dropout-based approaches [13] are also related but can be computationally expensive or unstable. In contrast, our method leverages training dynamics without modifying training or architecture. While training dynamics have been explored in snapshot ensembles [26] and dataset cartography [43], these works do not address selective prediction. Related efforts on example difficulty [1, 45] rely on costly gradient statistics or label access, while our method uses lightweight label trajectory disagreement signals. Finally, our approach builds on disagreement between intermediate predictions, a signal studied in the context of generalization and out-of-distribution detection [3, 29].

3. Our Method: Selective Prediction via Training Dynamics

We introduce a class of selective prediction methods based on training-time prediction instability, which we refer to as SPTD. The key intuition is that the evolution of a model's predictions during training encodes useful information about its eventual confidence and reliability. Rather than relying solely on the final model—as is common in existing selective prediction methods—we leverage the sequence of intermediate models generated during training. Let $[f_1, f_2, \ldots, f_T]$ be a sequence of intermediate checkpoints. We define a disagreement score at each time step t as a function $a_t : \mathcal{X} \to \mathbb{R}^+$ with $a_t(\boldsymbol{x}) = 0$ when $f_t(\boldsymbol{x}) = f_T(\boldsymbol{x})$. The form of a_t depends on the prediction task (e.g., classification vs regression), and we omit the dependence on $\boldsymbol{x}$ when clear from context.

Given a test point $\boldsymbol{x}$, our method computes a full disagreement sequence $[a_1, \ldots, a_T]$ over training checkpoints and weights each term by $v_t = (\frac{t}{T})^k$, where $k \in [0, \infty)$. This weighting penalizes later-stage disagreements more heavily, which is consistent with the observation that easier examples tend to converge earlier in training [43]. While prior work focused on convergence patterns of training examples, our key insight is that this behavior also extends to test inputs. The resulting selection function is a weighted sum of disagreements: $g(\boldsymbol{x}) = \sum_t v_t a_t(\boldsymbol{x})$, which captures the overall prediction instability of a test input across the training trajectory.

Instability for Classification. For classification, we define $a_t = 1 - \delta_{f_t(\boldsymbol{x}), f_T(\boldsymbol{x})}$, where δ is the Kronecker delta. Thus, $a_t = 1$ when the intermediate and final predictions disagree, and 0 otherwise. The classification-specific algorithm is shown in Algorithm 1. While continuous uncertainty measures (e.g., predictive entropy or softmax margin) are common, they tend to be noisy proxies. In contrast, discrete class disagreement offers a direct and reliable misclassification signal (see Appendix C for a discussion).

Algorithm 1 SPTD for classification

Require: Intermediate models $[f_1, \ldots, f_T]$, query point $\boldsymbol{x}$, weighting parameter $k \in [0, \infty)$.

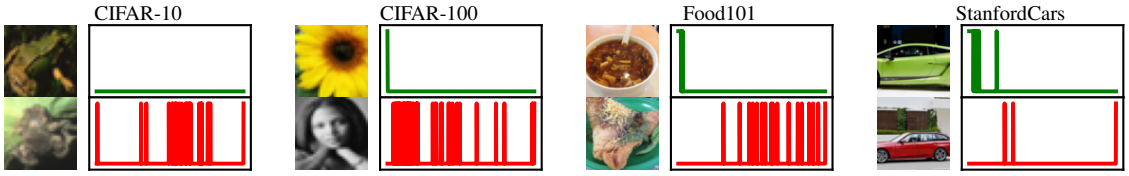
for $t \in [T]$ **do**
  if $f_t(\boldsymbol{x}) = f_T(\boldsymbol{x})$ **then** $a_t \leftarrow 0$ **else** $a_t \leftarrow 1$
  $v_t \leftarrow (\frac{t}{T})^k$
end for
$g \leftarrow \sum_t v_t a_t$
if $g \leq \tau$ **then** $f(\boldsymbol{x}) = f_T(\boldsymbol{x})$ **else** $f(\boldsymbol{x}) = \perp$

Instability for Regression. Our method naturally extends to regression. Here, we define the disagreement as the deviation from the final prediction: $a_t = \|f_t(\boldsymbol{x}) - f_T(\boldsymbol{x})\|$. The resulting procedure is identical to Algorithm 1, with only the definition of a_t changed (see Algorithm 2).

Instability for Time Series Prediction. For time series prediction, we generalize the regression formulation to each time step $r \in \{1, \ldots, R\}$, where $f_t(\boldsymbol{x}) \in \mathbb{R}^R$. We compute per-step disagreement as $a_{t,r} = \|f_t(\boldsymbol{x})_r - f_T(\boldsymbol{x})_r\|$ and define the selection function as the total instability across time: $g(\boldsymbol{x}) = \sum_r \sum_t v_t a_{t,r}(\boldsymbol{x})$. The resulting procedure mirrors Algorithm 2 and is detailed in Algorithm 3. While we apply this to time series forecasting, the formulation generalizes to any vector-valued prediction task where prediction variability is meaningful.

4. Empirical Evaluation

We present an empirical study demonstrating the effectiveness of SPTD across domains. Our results show that computing and thresholding the proposed weighted instability score from SPTD provides a strong score for selective classification, regression, and time series prediction.



(*a*) CIFAR-10 (*b*) CIFAR-100 (*c*) Food-101 (*d*) Stanford Cars

Figure 2: **Most characteristic examples across datasets**. For each dataset, we show the samples with the most stable and most unstable (dis-)agreement with the final label along with their corresponding a_t indicator function. Correct points are predominantly characterized by disagreements early in training while incorrect points change their class label throughout (but importantly close to the end of) training. We provide additional examples from all datasets in Figure 20 in the Appendix.

4.1. Classification

Datasets & Models. We evaluate SPTD on image dataset benchmarks that are common in the selective classification literature: CIFAR-10/CIFAR-100 [32], StanfordCars [31], and Food101 [7]. For each dataset, we train a deep neural network following the ResNet-18 architecture [23].

Baselines. We compare our method (SPTD) to common SC techniques previously introduced in Section 2: Softmax Response (SR), Self-Adaptive Training (SAT), and Deep Ensembles (DE) [33]. We do not include results for SelectiveNet, Deep Gamblers, or Monte-Carlo Dropout as previous works have shown that SAT strictly dominates these methods [12, 27]. Our hyper-parameter tuning procedure is documented in Appendix E.1.

Main Results. Consistent with standard selective classification evaluation, our main results examine the accuracy–coverage trade-off of SPTD. Table 1 compares SPTD to prior work, showing its effectiveness across the full coverage spectrum. We report results for SPTD, SAT, SR, and DE, observing that SPTD outperforms SAT and SR and performs comparably to DE. To further improve performance, we combine SPTD with DE by applying SPTD to each ensemble member and averaging the scores $\texttt{DE+SPTD} = \frac{1}{M}\sum_{m=1}^{M}\texttt{SPTD}_m$, where each $\texttt{SPTD}_m$ computes g for ensemble member $m \in [M]$. This yields new state-of-the-art results and demonstrates that SPTD can be flexibly integrated into existing pipelines.

Table 1: **Selective accuracy on CIFAR-100 (top) and StanfordCards (bottom).** Extended results in Table 3.

Coverage	SR	SAT	DE	SPTD	DE+SPTD
100	**75.1 (±0.0)**	**75.1 (±0.0)**	**75.1 (±0.0)**	**75.1 (±0.0)**	**75.1 (±0.0)**
90	78.2 (± 0.1)	78.9 (± 0.1)	80.2 (± 0.0)	80.4 (± 0.1)	**81.1 (± 0.1)**
80	82.1 (± 0.0)	82.9 (± 0.0)	84.7 (± 0.1)	84.6 (± 0.1)	**85.0 (± 0.2)**
70	86.4 (± 0.1)	87.2 (± 0.1)	88.6 (± 0.1)	**88.7 (± 0.0)**	**88.8 (± 0.1)**
60	90.0 (± 0.0)	90.3 (± 0.2)	90.2 (± 0.2)	90.1 (± 0.0)	**90.4 (± 0.1)**
50	92.9 (± 0.1)	93.3 (± 0.0)	94.8 (± 0.0)	94.6 (± 0.0)	**94.9 (± 0.0)**
40	95.1 (± 0.0)	95.2 (± 0.1)	**96.8 (± 0.1)**	**96.9 (± 0.1)**	**96.9 (± 0.0)**
30	97.2 (± 0.2)	97.5 (± 0.0)	**98.4 (± 0.1)**	**98.4 (± 0.1)**	**98.5 (± 0.0)**
20	97.8 (± 0.1)	98.3 (± 0.1)	**99.0 (± 0.0)**	98.8 (± 0.2)	**99.2 (± 0.1)**
10	98.1 (± 0.0)	98.8 (± 0.1)	99.2 (± 0.1)	**99.4 (± 0.1)**	**99.6 (± 0.1)**
100	**77.6 (±0.0)**	**77.6 (±0.0)**	**77.6 (±0.0)**	**77.6 (±0.0)**	**77.6 (±0.0)**
90	83.0 (±0.1)	83.0 (±0.2)	**83.7 (±0.1)**	83.3 (±0.1)	**83.7 (±0.2)**
80	87.6 (±0.0)	88.0 (±0.1)	88.7 (±0.1)	**89.3 (±0.0)**	**89.7 (±0.0)**
70	90.8 (±0.0)	92.2 (±0.1)	92.4 (±0.1)	**93.6 (±0.0)**	93.4 (±0.1)
60	93.5 (±0.1)	95.2 (±0.1)	95.3 (±0.0)	**96.2 (±0.0)**	**96.3 (±0.0)**
50	95.3 (±0.0)	96.9 (±0.2)	96.4 (±0.1)	**97.0 (±0.1)**	**97.1 (±0.3)**
40	96.8 (±0.0)	97.8 (±0.0)	**97.8 (±0.2)**	**97.8 (±0.1)**	**97.8 (±0.0)**
30	97.5 (±0.1)	98.2 (±0.2)	**98.6 (±0.0)**	98.2 (±0.2)	**98.9 (±0.0)**
20	98.1 (±0.0)	98.4 (±0.1)	**98.9 (±0.2)**	98.6 (±0.0)	**99.0 (±0.0)**
10	98.2 (±0.1)	98.7 (±0.1)	**99.5 (±0.1)**	98.5 (±0.1)	**99.5 (±0.0)**

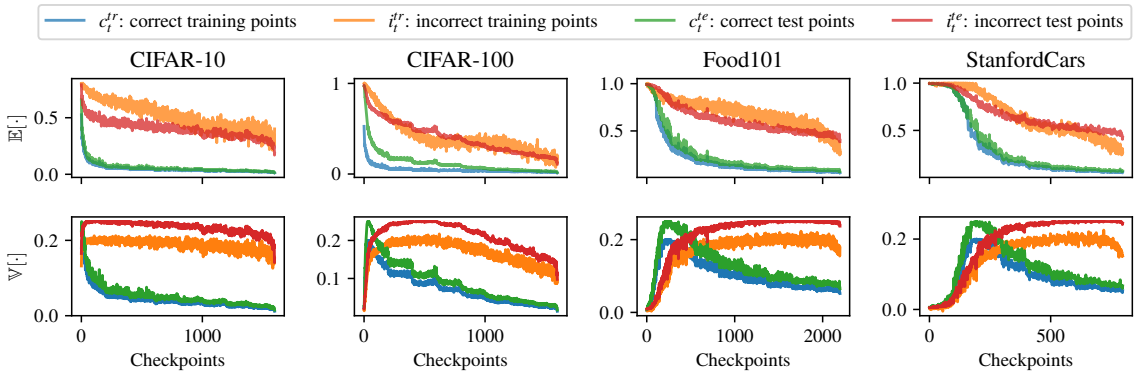


Figure 3: **Monitoring expectations $\mathbb{E}[\cdot]$ and variances $\mathbb{V}[\cdot]$ for correct/incorrect training and test points**. We observe that correctly classified points (cold colors) have both their expectations and variances quickly decreasing to 0 as training progresses. Incorrectly classified points (warm colors) both exhibit large expectations and variances and stay elevated over large periods.

Individual Evolution Plots. To evaluate our disagreement metric from Section 3, we track the evolution of the indicator variable a_t for individual test examples in Figure 2. For each dataset, we show the most stable and unstable points, observing that easy examples show minimal oscillation while harder ones fluctuate more, especially late in training. This aligns with our intuition: consistent predictions suggest correctness, while inconsistent ones indicate uncertainty. As shown in Figure 7 in the Appendix, our score $g(\cdot)$ also yields distinct distributions for correct vs. incorrect predictions, enabling effective thresholding for coverage/accuracy trade-offs.

Convergence Behavior of Training and Test Points. The effectiveness of `SPTD` relies on the hypothesis that correctly and incorrect predictions follow distinct training dynamics. Figure 3 supports this: expected disagreement for correctly classified training (c_t^{tr}) and test (c_t^{te}) points converges to zero during training, with decreasing variance. This indicates that correct predictions stabilize early, and fast convergence signals correctness. Harder datasets show slower convergence, reflecting greater optimization difficulty. In contrast, misclassified points (i_t^{tr}, i_t^{te}) maintain higher disagreement and variance. This clear separation underpins `SPTD`'s strong selective prediction performance.

4.2. Regression Experiments

Datasets & Models. Our experimental suite for regression considers the following datasets: California housing dataset [37], the concrete strength dataset [48], and the fish toxicity dataset [4]. We split all datasets into 80% training and 20% test sets after a random shuffle. Then, we train a fully connected neural network with layer dimensionalities $D \to 10 \to 7 \to 4 \to 1$.

Model Setup & Baselines. We consider the following baselines for rejecting input samples: (i) approximating the predictive variance using deep ensembles (`DE`) [33, 49]; (ii) SelectiveNet (`SN`) which explicitly optimizes utility given a desired coverage constraint; and (iii) training the model with a Gaussian parametric output distribution (`ODIST`) via maximum likelihood maximization [2].

Main results. We document our results in Figure 4. We see that `ODIST` delivers subpar results (likely due to mis-calibration) and does not provide a meaningful signal for selective prediction. On

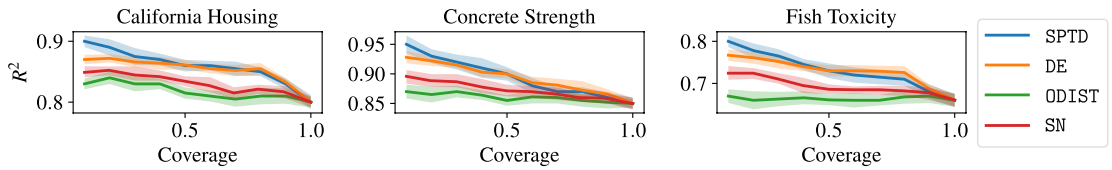


Figure 4: **R^2/coverage trade-off across various regression datasets**. SPTD offers comparable performance to DE but provides improved results at low coverage.

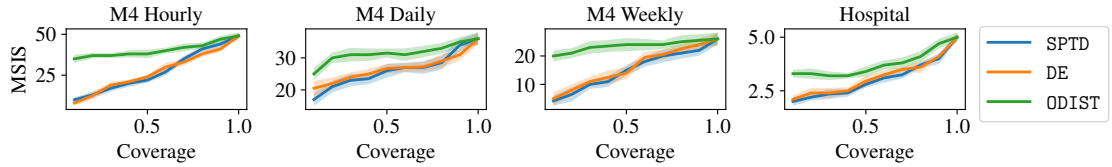


Figure 5: **MSIS/coverage trade-off across various time series prediction datasets**. SPTD offers comparable performance to DE but provides improved results at low coverage.

the other hand, DE and SPTD perform comparably, with SPTD consistently outperforming DE at low coverage across all benchmarked datasets.

4.3. Time Series Experiments

Datasets & Models. As part of our time series experiments, we consider the M4 forecasting competition dataset [35] which contains time series aggregated at various time intervals (e.g., hourly, daily). In addition, we also provide experimentation on the Hospital dataset [28]. Our experimentation is carried out using the GluonTS time series framework [2]. We carry out our experimentation using the DeepAR model [40], a recurrent neural network designed for time series forecasting.

Baselines. Our baselines correspond to those used for regression in Section 4.2: deep ensembles (DE), and output parameterization using a Student-t distribution (ODIST), which is expected to offer improved robustness to outliers (common in time series analysis) due to its heavier tails.

Main results. Our time series results are shown in Figure 5 and are consistent with our findings for regression: ODIST does not provide a meaningful signal for selective prediction, while SPTD and DE perform similarly well across the entire coverage spectrum.

5. Conclusion

In this work we have proposed SPTD, a selective prediction technique that relies on measuring prediction instability of test points over intermediate model states obtained during training. Our method offers several advantages over previous works. In particular (i) it can be applied to all existing models whose checkpoints were recorded (hence the potential for immediate impact); (ii) it is composable with existing selective prediction techniques; (iii) it can be readily applied to both discrete and real-valued prediction problems; and (iv) it is more computationally efficient than competing ensembling-based approaches. We verified the performance of SPTD using an extensive empirical evaluation, leading to new state-of-the-art performance in the field.

References

[1] Chirag Agarwal, Daniel D'souza, and Sara Hooker. Estimating example difficulty using variance of gradients. *arXiv preprint arXiv:2008.11600*, 2020.

[2] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, et al. Gluonts: Probabilistic time series models in python. *arXiv preprint arXiv:1906.05264*, 2019.

[3] Christina Baek, Yiding Jiang, Aditi Raghunathan, and J Zico Kolter. Agreement-on-the-line: Predicting the performance of neural networks under distribution shift. *Advances in Neural Information Processing Systems*, 35:19274–19289, 2022.

[4] Davide Ballabio, Matteo Cassotti, Viviana Consonni, and Roberto Todeschini. QSAR fish toxicity. UCI Machine Learning Repository, 2019. DOI: https://doi.org/10.24432/C5JG7B.

[5] Guy Bar-Shalom, Yonatan Geifman, and Ran El-Yaniv. Window-based distribution shift detection for deep neural networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[6] Raef Bassily, Vitaly Feldman, Cristóbal Guzmán, and Kunal Talwar. Stability of stochastic gradient descent on nonsmooth convex losses. *Advances in Neural Information Processing Systems*, 33:4381–4391, 2020.

[7] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

[8] Luís Felipe P Cattelan and Danilo Silva. Improving selective classification performance of deep neural networks through post-hoc logit normalization and temperature scaling. *arXiv preprint arXiv:2305.15508*, 2023.

[9] Robert Challen, Joshua Denny, Martin Pitt, Luke Gompels, Tom Edwards, and Krasimira Tsaneva-Atanasova. Artificial intelligence, bias and clinical safety. *BMJ Quality & Safety*, 28 (3):231–237, 2019.

[10] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. *Advances in Neural Information Processing Systems*, 30, 2017.

[11] Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(53):1605–1641, 2010. URL http://jmlr.org/papers/v11/el-yaniv10a.html.

[12] Leo Feng, Mohamed Osama Ahmed, Hossein Hajimirsadeghi, and Amir H Abdi. Towards better selective classification. In *The Eleventh International Conference on Learning Representations*, 2023.

[13] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[14] Ido Galil, Mohammed Dabbah, and Ran El-Yaniv. What can we learn from the selective prediction and uncertainty estimation performance of 523 imagenet classifiers. *arXiv preprint arXiv:2302.11874*, 2023.

[15] Aditya Gangrade, Anil Kag, and Venkatesh Saligrama. Selective classification via one-sided prediction. In *International Conference on Artificial Intelligence and Statistics*, pages 2179–2187. PMLR, 2021.

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

[17] Yonatan Geifman and Ran El-Yaniv. Selectivenet: A deep neural network with an integrated reject option. In *International Conference on Machine Learning*, pages 2151–2159. PMLR, 2019.

[18] Yonatan Geifman, Guy Uziel, and Ran El-Yaniv. Bias-reduced uncertainty estimation for deep neural classifiers. *arXiv preprint arXiv:1805.08206*, 2018.

[19] Zahra Ghodsi, Siva Kumar Sastry Hari, Iuri Frosio, Timothy Tsai, Alejandro Troccoli, Stephen W Keckler, Siddharth Garg, and Anima Anandkumar. Generating and characterizing scenarios for safety testing of autonomous vehicles. *arXiv preprint arXiv:2103.07403*, 2021.

[20] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

[21] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

[22] Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International conference on machine learning*, pages 1225–1234. PMLR, 2016.

[23] 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*, pages 770–778, 2016.

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

[25] Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. What do compressed deep neural networks forget? *arXiv preprint arXiv:1911.05248*, 2019.

[26] 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*, 2017.

[27] Lang Huang, Chao Zhang, and Hongyang Zhang. Self-adaptive training: beyond empirical risk minimization. *Advances in neural information processing systems*, 33:19365–19376, 2020.

[28] RJ Hyndman. expsmooth: Data sets from "forecasting with exponential smoothing". *R package version*, 2, 2015.

[29] Yiding Jiang, Vaishnavh Nagarajan, Christina Baek, and J Zico Kolter. Assessing generalization of sgd via disagreement. *arXiv preprint arXiv:2106.13799*, 2021.

[30] Ziheng Jiang, Chiyuan Zhang, Kunal Talwar, and Michael C Mozer. Characterizing structural regularities of labeled data in overparameterized models. *arXiv preprint arXiv:2002.03206*, 2020.

[31] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013.

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

[33] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf.

[34] Ziyin Liu, Zhikang Wang, Paul Pu Liang, Russ R Salakhutdinov, Louis-Philippe Morency, and Masahito Ueda. Deep gamblers: Learning to abstain with portfolio theory. *Advances in Neural Information Processing Systems*, 32, 2019.

[35] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1): 54–74, 2020.

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

[37] R Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

[38] Stephan Rabanser, Anvith Thudi, Abhradeep Thakurta, Krishnamurthy Dvijotham, and Nicolas Papernot. Training private models that know what they don't know. *arXiv preprint arXiv:2305.18393*, 2023.

[39] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International conference on machine learning*, pages 4393–4402. PMLR, 2018.

[40] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36 (3):1181–1191, 2020.

[41] Sara Sangalli, Ertunc Erdil, and Ender Konukoglu. Expert load matters: operating networks at high accuracy and low manual effort. *Advances in Neural Information Processing Systems*, 36, 2024.

[42] Andrew I Schein and Lyle H Ungar. Active learning for logistic regression: an evaluation. *Machine Learning*, 68(3):235–265, 2007.

[43] Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics. *arXiv preprint arXiv:2009.10795*, 2020.

[44] Anvith Thudi, Hengrui Jia, Ilia Shumailov, and Nicolas Papernot. On the necessity of auditable algorithmic definitions for machine unlearning. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4007–4022, 2022.

[45] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

[46] Lucas Nunes Vieira, Minako O'Hagan, and Carol O'Sullivan. Understanding the societal impacts of machine translation: a critical review of the literature on medical and legal use cases. *Information, Communication & Society*, 24(11):1515–1532, 2021.

[47] Yu-Chang Wu, Shen-Huan Lyu, Haopu Shang, Xiangyu Wang, and Chao Qian. Confidence-aware contrastive learning for selective classification. *arXiv preprint arXiv:2406.04745*, 2024.

[48] I-Cheng Yeh. Concrete Compressive Strength. UCI Machine Learning Repository, 2007. DOI: https://doi.org/10.24432/C5PK67.

[49] Ahmed Zaoui, Christophe Denis, and Mohamed Hebiri. Regression with reject option and application to knn. *Advances in Neural Information Processing Systems*, 33:20073–20082, 2020.

Appendix A. Utility Functions For Different Prediction Tasks

We define the following utility functions to be used based on the problem domain:

1. *Classification*: We use accuracy on accepted points as our utility function for classification:

$$\text{Accuracy} = \frac{1}{M_\tau} \sum_{i=1}^{M_\tau} \mathbb{1}[\boldsymbol{x}_i : f(\boldsymbol{x}_i) = y_i] \tag{2}$$

2. *Regression*: We use the coefficient of determination (R^2 score, which is a scaled version of the mean squared error) on accepted points as our utility function for regression:

$$R^2 = 1 - \frac{\sum_{i=1}^{M_\tau} (f(\boldsymbol{x}_i) - y_i)^2}{\sum_{i=1}^{M_\tau} (y_i - \frac{1}{M_\tau} \sum_{j=1}^{M_\tau} y_j)^2} \tag{3}$$

3. *Time Series Forecasting*: We use the Mean Scaled Interval Score (MSIS) [20] on accepted series as our utility function for time series forecasting

$$\text{MSIS} = \frac{1}{M_\tau R} \sum_{i=1}^{M_\tau} \frac{\begin{array}{r}\sum_{r=n+1}^{n+R} (u_{i,r} - l_{i,r}) + \frac{2}{\alpha} (l_{i,r} - y_{i,r}) \mathbb{1}[y_{i,r} < l_{i,r}] \\ + \frac{2}{\alpha} (y_{i,r} - u_{i,r}) \mathbb{1}[y_{i,r} > u_{i,r}]\end{array}}{\frac{1}{n-m} \sum_{r=m+1}^{n} |y_{i,r} - y_{i,r-m}|} \tag{4}$$

   where α refers to a specific predictive quantile, n to the conditioning length of the time series, m to the length of the seasonal period, and $u_{i,r}$ and $l_{i,r}$ to the upper and lower bounds on the prediction range, respectively.

Appendix B. Method Intuition: Prediction Disagreements Generalize Softmax Response

Stochastic iterative optimization procedures, such as Stochastic Gradient Descent (SGD), yield a sequence of models that is iteratively derived by minimizing a loss function $\ell(\cdot, \cdot)$ on a randomly selected mini-batch $(\boldsymbol{X}_i, \boldsymbol{y}_i)$ from the training set. The iterative update rule can be expressed as follows:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \nu \frac{\partial \ell(f(\boldsymbol{X}_i), \boldsymbol{y}_i)}{\partial \boldsymbol{\theta}_t} \tag{5}$$

where the learning rate ν controls the speed of optimization and $t \in \{1, \ldots, T\}$ represents a particular time-step during the optimization process. Current methods for selective prediction disregard the properties of this iterative process and only rely on the final set of parameters $\boldsymbol{\theta}_T$. However, the optimization trajectories contain information that we can use to determine prediction reliability. In particular, on hard optimization tasks, the presence of stochasticity from SGD and the potential ambiguity of the data often leads to noisy optimization behavior. As a result, intermediate predictions produced over the course of training might widely disagree in what the right prediction would be for a given data point. Our class of selective prediction approaches explicitly make use of these training dynamics by formalizing rejection scores based on the observed frequency of prediction disagreements with the final model throughout training.

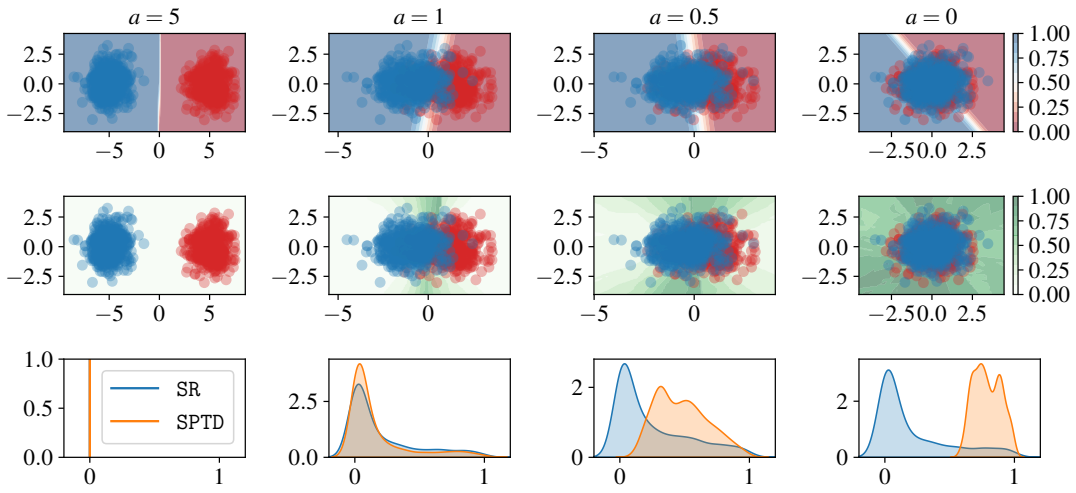


Figure 6: **Synthetic example of anomaly scoring based on `SR` vs `SPTD`**. The first row shows a test set from the generative Gaussian model as well as the learned decision boundary separating the two Gaussians. For small a, the decision boundary is overconfident. The second row shows the same data set but instead depicts the scores yielded by applying `SPTD` to the full domain. `SPTD` highlights rightful rejection regions more clearly than the overconfident `SR` score: larger regions are flagged as exhibiting noisy training dynamics (with stronger shades of green indicating stronger disagreement) as $a \to 0$. The bottom row shows the distribution of the `SR` and `SPTD` scores, clearly showing that `SPTD` leads to improved uncertainty under stronger label ambiguity.

To illustrate and reinforce this intuition that training dynamics contain meaningfully more useful information for selective prediction than the final model, we present a synthetic logistic regression example. First, we generate a mixture of two Gaussians each consisting of 1000 samples: $D = \{(\boldsymbol{x}_i, 0)\}_{i=1}^{1000} \cup \{(\boldsymbol{x}_j, 1)\}_{j=1}^{1000}$ where $\boldsymbol{x}_i \sim \mathcal{N}(\begin{bmatrix} a & 0 \end{bmatrix}^\top, \boldsymbol{I})$ and $\boldsymbol{x}_j \sim \mathcal{N}(\begin{bmatrix} -a & 0 \end{bmatrix}^\top, \boldsymbol{I})$. Note that a controls the distance between the two 2-dimensional Gaussian clusters, allowing us to specify the difficulty of the learning task. Then, we train a linear classification model using SGD for 1000 epochs for each $a \in \{0, 0.5, 1, 5\}$. Finally, we compute both the softmax response score (`SR`) score, the typical baseline for selective classification, as well as our `SPTD` score (details in Section **??**).

We showcase the results from this experiment in Figure 6. We see that if the data is linearly separable ($a = 5$) the learned decision boundary is optimal and the classifier's built-in confidence score `SR` reflects well-calibrated uncertainty. Moreover, the optimization process is stable as `SPTD` yields low scores over the full domain. However, as we move the two Gaussians closer together (i.e., by reducing a) we see that the `SR` baseline increasingly suffers from overconfidence: large parts of the domain are very confidently classified as either 0 (red) or 1 (blue) with only a small ambiguous region (white). However, the optimization trajectory is highly unstable with the decision boundary changing abruptly between successive optimization steps. `SPTD` identifies the region of datapoints exhibiting large prediction disagreement due to this training instability and correctly rejects them (as those are regions also subject to label ambiguity in this case). In summary, we observe that `SPTD` provides improved uncertainty quantification in ambiguous classification regions (which induce training instability) and reduces to the `SR` solution as the classification task becomes

Algorithm 2 SPTD for regression

Require: Intermediate models $[f_1, \ldots, f_T]$, query point $\boldsymbol{x}$, weighting parameter $k \in [0, \infty)$.

for $t \in [T]$ **do**
  $a_t \leftarrow ||f_t(\boldsymbol{x}) - f_T(\boldsymbol{x})||$
  $v_t \leftarrow (\frac{t}{T})^k$
end for
$g \leftarrow \sum_t v_t a_t$
if $g \leq \tau$ **then** $f(\boldsymbol{x}) = f_T(\boldsymbol{x})$ **else** $f(\boldsymbol{x}) = \perp$

Algorithm 3 SPTD for time series forecasting

Require: Intermediate models $[f_1, \ldots, f_T]$, query point $\boldsymbol{x}$, weighting $k \in [0, \infty)$, prediction horizon R.

for $t \in [T]$ **do**
  for $r \in [R]$ **do**
    $a_{t,r} \leftarrow ||f_t(\boldsymbol{x})_r - f_T(\boldsymbol{x})_r||$
  end for
  $v_t \leftarrow (\frac{t}{T})^k$
end for
$g \leftarrow \sum_r \sum_t v_t a_{t,r}$
if $g \leq \tau$ **then** $f(\boldsymbol{x}) = L$ **else** $f(\boldsymbol{x}) = \perp$

easier. Hence, we expect SPTD to generalize SR performance, which is supported by this logistic regression experiment.

Appendix C. Alternate Metric Choices

We briefly discuss additional potential metric choices that we investigated but which lead to selective classification performance worse than our main method.

C.1. Jump Score s_{jmp}

We also consider a score which captures the level of disagreement between the predicted label of two successive intermediate models (i.e., how much jumping occurred over the course of training). For $j_t = 0$ iff $f_t(\boldsymbol{x}) = f_{t-1}(\boldsymbol{x})$ and $j_t = 1$ otherwise we can compute the jump score as $s_{\text{jmp}} = 1 - \sum v_t j_t$ and threshold it as in § D.2 and § D.3.

C.2. Variance Score s_{var} for Continuous Metrics

We consider monitoring the evolution of continuous metrics that have been shown to be correlated with example difficulty. These metrics include (but are not limited to):

- Confidence (conf): $\max_{c \in \mathcal{Y}} f_t(\boldsymbol{x})$
- Confidence gap between top 2 most confident classes (gap): $\max_{c \in \mathcal{Y}} f_t(\boldsymbol{x}) - \max_{c \neq \hat{y}} f_t(\boldsymbol{x})$
- Entropy (ent): $-\sum_{c=1}^{C} f_t(\boldsymbol{x})_c \log\left(f_t(\boldsymbol{x})_c\right)$

[30] show that example difficulty is correlated with confidence and entropy. Moreover, they find that difficult examples are learned later in the training process. This observation motivates designing a score based on these continuous metrics that penalises changes later in the training process more heavily. We consider the maximum softmax class probability known as confidence, the negative entropy and the gap between the most confident classes for each example instead of the model predictions. Assume that any of these metrics is given by a sequence $z = \{z_1, \ldots, z_T\}$ obtained from T intermediate models. Then we can capture the uniformity of z via a (weighted) variance score $s_{\text{var}} = \sum_t w_t(z_t - \mu)^2$ for mean $\mu = \frac{1}{T}\sum_t z_t$ and an increasing weighting sequence $w = \{w_1, \ldots, w_T\}$.

In order to show the effectiveness of the variance score s_{var} for continuous metrics, we provide a simple bound on the variance of confidence $\max_{y \in \mathcal{Y}} f_t(\boldsymbol{x})$ in the final checkpoints of the training. Assuming that the model has converged to a local minima with a low learning rate, we can assume that the distribution of model weights can be approximated by a Gaussian distribution.

We consider a linear regression problem where the inputs are linearly separable.

Lemma 1 *Assume that we have some Gaussian prior on the model parameters in the logistic regression setting across m final checkpoints. More specifically, given T total checkpoints of model parameters $\{\boldsymbol{w}_1, \boldsymbol{w_2}, \ldots, \boldsymbol{w}_T\}$ we have $p(W = \boldsymbol{w}_t) = \mathcal{N}(\boldsymbol{w}_0 \mid \boldsymbol{\mu}, s\boldsymbol{I})$ for $t \in \{T-m+1, \ldots, T\}$ and we assume that final checkpoints of the model are sampled from this distribution. We show that the variance of model confidence $\max_{y \in \{-1,1\}} p(y \mid \boldsymbol{x}_i, \boldsymbol{w}_t)$ for a datapoint $(\boldsymbol{x}_i, y_i)$ can be upper bounded by a factor of probability of correctly classifying this example by the optimal weights.*

Proof We first compute the variance of model predictions $p(y_i \mid \boldsymbol{x}_i, W)$ for a given datapoint $(\boldsymbol{x}_i, y_i)$. Following previous work [10, 42], the variance of predictions over these checkpoints can be estimated as follows:

Taking two terms in Taylor expansion for model predictions we have $p(y_i \mid \boldsymbol{x}_i, W) \simeq p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}) + g_i(\boldsymbol{w})^\top (W - \boldsymbol{w})$ where W and $\boldsymbol{w}$ are current and the expected estimate of the parameters and $g_i(\boldsymbol{w}) = p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w})(1 - p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}))\boldsymbol{x}_i$ is the gradient vector. Now we can write the variance with respect to the model prior as:

$$\mathbb{V}\left(p(y_i \mid \boldsymbol{x}_i, W)\right) \simeq \mathbb{V}\left(g_i(\boldsymbol{w})^\top (W - \boldsymbol{w})\right) = g_i(\boldsymbol{w})^\top F^{-1} g_i(\boldsymbol{w})$$

where F is the variance of posterior distribution $p(W \mid X, Y) \sim \mathcal{N}(W \mid \boldsymbol{w}, F^{-1})$. This suggests that the variance of probability of correctly classifying $\boldsymbol{x}_i$ is proportional to $p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w})^2(1 - p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}))^2$. Now we can bound the variance of maximum class probability or confidence as below:

$$\begin{aligned}\mathbb{V}\left(\max_{y \in \{-1,1\}} p(y \mid \boldsymbol{x}_i, W)\right) &\leq \mathbb{V}\left(p(y_i \mid \boldsymbol{x}_i, W)\right) + \mathbb{V}\left(p(-y_i \mid \boldsymbol{x}_i, W)\right) \\ &\approx 2p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w})^2(1 - p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}))^2 \boldsymbol{x}_i^\top F^{-1} \boldsymbol{x}_i\end{aligned}$$

■

We showed that if the probability of correctly classifying an example given the final estimate of model parameters is close to one, the variance of model predictions following a Gaussian prior gets close to zero, we expect a similar behaviour for the variance of confidence under samples of this distribution.

Appendix D. Selective Prediction and Forging

While our SPTD method is primarily motivated from the example difficulty view point, we remark that the scores SPTD computes to decide which points to reject can be derived from multiple different perspectives. To showcase this, we provide a formal treatment on the connection between selective classification and forging [44], which ultimately leads to the same selection function $g(\cdot)$ as above.

Previous work has shown that running SGD on different datasets could lead to the same final model [6, 22, 44]. For example, this is intuitive when two datasets were sampled from the same distribution. We would then expect that training on either dataset should not significantly affect the model returned by SGD. For our selective prediction problem, this suggests an approach to decide which points the model is likely to predict correctly on: identify the datasets that it could have been trained on (in lieu of the training set it was actually trained on). Any point from the datasets the model could have trained on would then be likely to be predicted on correctly by the model. Recent work on forging [44] solves this problem of identifying datasets the model could have trained on by brute-force searching through different mini-batches to determine if a mini-batch in the alternative dataset can be used to reproduce one of the original training steps. Even then, this is only a sufficient condition to show a datapoint could have plausibly been used to train: if the brute-force fails, it does not mean the datapoint could not have been used to obtain the final model. As an alternative, we propose to instead characterize the optimization behaviour of training on a dataset as a probabilistic necessary condition, i.e, a condition most datapoints that were (plausibly) trained on would satisfy based on training dynamics. Our modified hypothesis is then that the set of datapoints we optimized for (which contains the forgeable points) coincides significantly with the set of points the model predicts correctly on.

D.1. A Framework for Being Optimized

In this section we derive an upper-bound on the probability that a datapoint could have been used to obtain the model's checkpointing sequence. This yields a probabilistically necessary (though not sufficient) characterization of the points we explicitly optimized for. This bound, and the variables it depends on, informs what we characterize as "optimizing" for a datapoint, and, hence, our selective classification methods.

Let us denote the set of all datapoints as $\mathcal{D}$, and let $D \subset \mathcal{D}$ be the training set. We are interested in the setting where a model f is plausibly sequentially trained on D (e.g., with stochastic gradient descent). We thus also have access to a sequence of T intermediate states for f, which we denote $[f_1, \ldots, f_T]$. In this sequence, note that f_T is exactly the final model f.

Now, let p_t represent the random variable for outputs on D given by an intermediate model f_t where the outputs have been binarized: we have 0 if the output agrees with the final prediction and 1 if not. In other words, p_t is the distribution of labels given by first drawing $\boldsymbol{x} \sim D$ and then outputting $1 - \delta_{f_t(\boldsymbol{x}), f_T(\boldsymbol{x})}$ where δ denotes the Dirac delta function. Note that we always have both a well-defined mean and variance for p_t as it is bounded. Furthermore, we always have the variances and expectations of $\{p_t\}$ converge to 0 with increasing t: as $p_T = 0$ always and the sequence is finite convergence trivially occurs. To state this formally, let $v_t = \mathbb{V}_{\boldsymbol{x} \sim D}[p_t]$ and let $e_t = \mathbb{E}_{\boldsymbol{x} \sim D}[p_t]$ denote the variances and expectations over points in D. In particular, we remark that $e_T = 0,\ v_T = 0$, so both e_t and v_t converge. More formally, for all $\epsilon > 0$ there exists an

$N \in \{1, \ldots, T\}$ such that $v_t < \epsilon$ for all $t > N$. Similarly, for all $\epsilon > 0$ there exists a (possibly different) $N \in \{1, \ldots, T\}$ such that $e_t < \epsilon$ for all $t > N$.

However, the core problem is that we do not know how this convergence in the variance and expectation occurs. More specifically, if we knew the exact values of e_t and v_t, we could use the following bound on the fraction of training data points producing a given $[a_1, \cdots, a_t]$ as a reject option for points that are not optimized for. We consequently introduce the notation $[a_1, \ldots, a_T]$ where $a_t = 1 - \delta_{f_t(\boldsymbol{x}), f_T(\boldsymbol{x})}$ which we call the "label disagreement (at t)". Note that the a_t are defined with respect to a given input, while p_t represent the distribution of a_t over all inputs in D.

Lemma 2 *Given a datapoint $\boldsymbol{x}$, let $\{a_1, \ldots, a_T\}$ where $a_t = 1 - \delta_{f_t(\boldsymbol{x}), f_T(\boldsymbol{x})}$. Assuming not all $a_t = e_t$ then the probability $\boldsymbol{x} \in D$ is $\leq \min_{v_t \ s.t \ a_t \neq e_t} \frac{v_t}{|a_t - e_t|^2}$.*

Proof By Chebyshev's inequality we have the probability of a particular sequence $\{a_1, \ldots, a_T\}$ occurring for a training point is $\leq \frac{v_t}{|a_t - e_t|^2}$ for every t (a bound on any of the individual a_t occurring as that event is in the event $|p_t - e_t| \geq |a_t - e_t|$ occurs). By taking the minimum over all these upper-bounds we obtain our upper-bound. ■

We do not guarantee Lemma 2 is tight. Though we do take a minimum to make it tighter, this is a minimum over inequalities all derived from Chebyshev's inequality[1]. Despite this potential looseness, using the bound from Lemma 2, we can design a naïve selective classification protocol based on the "optimized = correct (often)" hypothesis and use the above bound on being a plausible training datapoint as our characterization of optimization; for a test input $\boldsymbol{x}$, if the upper-bound on the probability of being a datapoint in D is lower than some threshold τ reject, else accept. However, the following question prevents us from readily using this method: *How do $\mathbb{E}[p_t]$ and $\mathbb{V}[p_t]$ evolve during training?*

To answer this question, we propose to examine how the predictions on plausible training points evolve during training. Informally, the evolution of $\mathbb{E}[p_t]$ represents knowing how often we predict the final label at step t, while the evolution of $\mathbb{V}[p_t]$ represents knowing how we become more consistent as we continue training. Do note that the performance of this optimization-based approach to selective classification will depend on how unoptimized incorrect test points are. In particular, our hypothesis is that incorrect points often appear sufficiently un-optimized, yielding distinguishable patterns for $\mathbb{E}[p_t]$ and $\mathbb{V}[p_t]$ when compared to optimized points. We verify this behavior in Section 4 where we discuss the distinctive label evolution patterns of explicitly optimized, correct, and incorrect datapoints.

D.2. Last Disagreement Model Score For Discrete Prediction (s_{MAX})

Here, we propose a selective classification approach based on characterizing optimizing for a datapoint based off of Lemma 2. Recall the bound given in Lemma 2 depends on expected values and variances for the p_t (denoted e_t and v_t respectively). In Section 4 we observe that e_t quickly converge to 0, and so by assuming $e_t = 0$ always[2] the frequentist bound on how likely a datapoint is a training point becomes $\min_{t \ s.t \ a_t = 1} \frac{v_t}{|a_t - e_t|^2} = \min_{t \ s.t \ a_t = 1} v_t$. Using this result for selective classification, we would impose acceptance if $\min_{t \ s.t \ a_t = 1} v_t \geq \tau$. Moreover, in Section 4, we further observe that v_t monotonically decreases in a convex manner (after an initial burn-in phase). Hence,

1. One could potentially use information about the distribution of points not in D to refine this bound.
2. We tried removing this assumption and observed similar performance.

imposing $\min_{t\ s.t\ a_t=1} v_t \geq \tau$ simply imposes a last checkpoint that can have a disagreement with the final prediction.

Based on these insights, we propose the following selective classification score: $s_{\max} = \max_{t\ s.t\ a_t=1} \frac{1}{v_t}$. Note that this score directly follows from the previous discussion but flips the thresholding direction from $\min_{t\ s.t\ a_t=1} v_t \geq \tau$ to $\max_{t\ s.t\ a_t=1} \frac{1}{v_t} \leq \tau$ for consistency with the anomaly scoring literature [39]. Finally, we choose to approximate the empirical trend of v_t as observed in Section 4 with $v_t = 1 - t^k$ for $k \in [1, \infty)$. Based on the choice of k, this approximation allows us to (i) avoid explicit estimation of v_t from validation data; and (ii) enables us to flexibly specify how strongly we penalize model disagreements late in training.

Hence, our first algorithm for selective classification is:

1. Denote $L = f_T(\boldsymbol{x})$, i.e. the label our final model predicts.
2. If $\exists t\ s.t\ a_t = 1$ then compute $s_{\max} = \max_{t\ s.t\ a_t=1} \frac{1}{v_t}$ as per the notation in Section D.1 (i.e $a_t = 1$ iff $f_t(x) \neq L$), else accept $\boldsymbol{x}$ with prediction L.
3. If $s_{\max} \leq \tau$ accept $\boldsymbol{x}$ with prediction L, else reject ($\perp$).

Note once again, as all our candidate $\frac{1}{v_t}$ increase, the algorithm imposes a last intermediate model which can output a prediction that disagrees with the final prediction: hereafter, the algorithm must output models that consistently agree with the final prediction.

D.3. Overall Disagreement Model Score (s_{SUM})

Note that the previous characterization of optimization, defined by the score s_{MAX}, could be sensitive to stochasticity in training and hence perform sub-optimally. That is, the exact time of the last disagreement, which s_{MAX} relies on, is subject to high noise across randomized training runs. In light of this potential limitation we propose the following "summation" algorithm which computes a weighted sum over training-time disagreements to get a more consistent statistic. Do note that typically to get a lower-variance statistic one would take an average, but multiplying by scalars can be replaced by correspondingly scaling the threshold we use. Hence, our proposed algorithm is:

1. Denote $L = f_T(\boldsymbol{x})$, i.e. the label our final model predicts.
2. If $\exists t\ s.t\ a_t = 1$, compute $s_{\text{sum}} = \sum_{t=1}^{T} \frac{a_t}{v_t}$, else accept $\boldsymbol{x}$ with prediction L.
3. If $s_{\text{sum}} \leq \tau$ accept $\boldsymbol{x}$ with prediction L, else reject ($\perp$).

Recalling our previous candidates for v_t, we have the s_{SUM} places higher weight on late disagreements. This gives us a biased average of the disagreements which intuitively approximates the expected last disagreement but now is less susceptible to noise. More generally, this statistic allows us to perform selective classification by utilizing information from all the disagreements during training. In Appendix E.2.8, we experimentally show that s_{SUM} leads to more robust selective classification results compared to s_{MAX}. **We remark that the sum score s_{SUM} corresponds exactly to our score $g(\cdot)$ proposed as part of `SPTD`, showcasing the strong connection of our method to forging.**

Appendix E. Extension of Empirical Evaluation

E.1. Full Hyper-Parameters

We document full hyper-parameter settings for our method (SPTD) as well as all baseline approaches in Table 2.

Table 2: **Hyper-parameters used for all algorithms for classification.**

Dataset	SC Algorithm	Hyper-Parameters
CIFAR-10	Softmax Response (SR)	N/A
	Self-Adaptive Training (SAT)	$P = 100$
	Deep Ensembles (DE)	$E = 10$
	Selective Prediction Training Dynamics (SPTD)	$T = 1600,\ k = 2$
CIFAR-100	Softmax Response (SR)	N/A
	Self-Adaptive Training (SAT)	$P = 100$
	Deep Ensembles (DE)	$E = 10$
	Selective Prediction Training Dynamics (SPTD)	$T = 1600,\ k = 2$
Food101	Softmax Response (SR)	N/A
	Self-Adaptive Training (SAT)	$P = 100$
	Deep Ensembles (DE)	$E = 10$
	Selective Prediction Training Dynamics (SPTD)	$T = 2200,\ k = 3$
StanfordCars	Softmax Response (SR)	N/A
	Self-Adaptive Training (SAT)	$P = 100$
	Deep Ensembles (DE)	$E = 10$
	Selective Prediction Training Dynamics (SPTD)	$T = 800,\ k = 5$

E.2. Additional Selective Prediction Results

E.2.1. CLASSIFICATION

Full Results Table. We show our full results in Table 7.

Checkpoint Weighting Sensitivity. One important hyper-parameter of our method is the weighting of intermediate predictions. Recall from Section 3 that SPTD approximates the expected stability for correctly classified points via a weighting function $v_t = (\frac{t}{T})^k$. In Figure 8, we observe that SPTD is robust to the choice of k and that $k \in [1, 3]$ performs best. At the same time, we find that increasing k too much leads to a decrease in accuracy at medium coverage levels. This result emphasizes that (i) large parts of the training process contain valuable signals for selective classification; and that (ii) early label disagreements arising at the start of optimization should be de-emphasized by our method.

Checkpoint Selection Strategy. The second important hyper-parameter of our method is the checkpoint selection strategy. In particular, to reduce computational cost, we study the sensitivity of SPTD with respect to the checkpointing resolution in Figure 9. Our experiments demonstrate

	Coverage	SR	SAT	DE	SPTD	DE+SPTD
CIFAR-10	100	**92.9 (±0.0)**	**92.9 (±0.0)**	**92.9 (±0.0)**	**92.9 (±0.0)**	**92.9 (±0.1)**
	90	96.4 (±0.1)	96.3 (±0.1)	**96.8 (±0.1)**	96.5 (±0.0)	**96.7 (±0.1)**
	80	98.1 (±0.1)	98.1 (±0.1)	**98.7 (±0.0)**	98.4 (±0.1)	**98.8 (±0.1)**
	70	98.6 (±0.2)	99.0 (±0.1)	**99.4 (±0.1)**	99.2 (±0.0)	**99.5 (±0.0)**
	60	98.7 (±0.1)	99.4 (±0.0)	99.6 (±0.1)	**99.6 (±0.2)**	**99.8 (±0.0)**
	50	98.6 (±0.2)	99.7 (±0.1)	99.7 (±0.1)	99.8 (±0.0)	**99.9 (±0.0)**
	40	98.7 (±0.0)	99.7 (±0.0)	99.8 (±0.0)	99.8 (±0.1)	**100.0 (±0.0)**
	30	98.5 (±0.0)	99.8 (±0.0)	99.8 (±0.0)	99.8 (±0.1)	**100.0 (±0.0)**
	20	98.5 (±0.1)	99.8 (±0.1)	99.8 (±0.0)	**100.0 (±0.0)**	**100.0 (±0.0)**
	10	98.7 (±0.0)	99.8 (±0.1)	99.8 (±0.1)	**100.0 (±0.0)**	**100.0 (±0.0)**
CIFAR-100	100	**75.1 (±0.0)**	**75.1 (±0.0)**	**75.1 (±0.0)**	**75.1 (±0.0)**	**75.1 (±0.0)**
	90	78.2 (± 0.1)	78.9 (± 0.1)	80.2 (± 0.0)	80.4 (± 0.1)	**81.1 (± 0.1)**
	80	82.1 (± 0.0)	82.9 (± 0.0)	84.7 (± 0.1)	84.6 (± 0.1)	**85.0 (± 0.2)**
	70	86.4 (± 0.1)	87.2 (± 0.1)	88.6 (± 0.1)	**88.7 (± 0.0)**	**88.8 (± 0.1)**
	60	90.0 (± 0.0)	90.3 (± 0.2)	90.2 (± 0.2)	90.1 (± 0.0)	**90.4 (± 0.1)**
	50	92.9 (± 0.1)	93.3 (± 0.0)	94.8 (± 0.0)	94.6 (± 0.0)	**94.9 (± 0.0)**
	40	95.1 (± 0.0)	95.2 (± 0.1)	**96.8 (± 0.1)**	**96.9 (± 0.1)**	**96.9 (± 0.0)**
	30	97.2 (± 0.2)	97.5 (± 0.0)	**98.4 (± 0.1)**	**98.4 (± 0.1)**	**98.5 (± 0.0)**
	20	97.8 (± 0.1)	98.3 (± 0.1)	**99.0 (± 0.0)**	98.8 (± 0.2)	**99.2 (± 0.1)**
	10	98.1 (± 0.0)	98.8 (± 0.1)	99.2 (± 0.1)	**99.4 (± 0.1)**	**99.6 (± 0.1)**
Food101	100	**81.1 (±0.0)**	**81.1 (±0.0)**	**81.1 (±0.0)**	**81.1 (±0.0)**	**81.1 (±0.0)**
	90	85.3 (±0.1)	85.5 (±0.2)	86.2 (±0.1)	85.7 (±0.0)	**86.7 (±0.0)**
	80	87.1 (±0.0)	89.5 (±0.0)	90.3 (±0.0)	89.9 (±0.0)	**91.3 (±0.1)**
	70	92.1 (±0.1)	92.8 (±0.1)	**94.5 (±0.1)**	93.7 (±0.0)	**94.6 (±0.0)**
	60	95.2 (±0.1)	95.5 (±0.1)	**97.0 (±0.0)**	**97.0 (±0.0)**	**97.0 (±0.0)**
	50	97.3 (±0.1)	97.5 (±0.0)	98.2 (±0.0)	**98.3 (±0.2)**	**98.5 (±0.0)**
	40	98.7 (±0.0)	98.7 (±0.2)	**99.1 (±0.0)**	99.1 (±0.1)	**99.2 (±0.1)**
	30	99.5 (±0.0)	99.7 (±0.2)	99.2 (±0.0)	99.6 (±0.0)	**99.7 (±0.0)**
	20	99.7 (±0.1)	99.7 (±0.2)	**99.9 (±0.1)**	**99.8 (±0.0)**	**99.9 (±0.1)**
	10	99.8 (±0.0)	99.8 (±0.1)	**99.9 (±0.1)**	**99.9 (±0.1)**	**99.9 (±0.1)**
StanfordCars	100	**77.6 (±0.0)**	**77.6 (±0.0)**	**77.6 (±0.0)**	**77.6 (±0.0)**	**77.6 (±0.0)**
	90	83.0 (±0.1)	83.0 (±0.2)	**83.7 (±0.1)**	83.3 (±0.1)	**83.7 (±0.2)**
	80	87.6 (±0.0)	88.0 (±0.1)	88.7 (±0.1)	**89.3 (±0.0)**	**89.7 (±0.0)**
	70	90.8 (±0.0)	92.2 (±0.1)	92.4 (±0.1)	**93.6 (±0.0)**	93.4 (±0.1)
	60	93.5 (±0.1)	95.2 (±0.1)	95.3 (±0.0)	**96.2 (±0.0)**	**96.3 (±0.0)**
	50	95.3 (±0.0)	96.9 (±0.2)	96.4 (±0.1)	**97.0 (±0.1)**	**97.1 (±0.3)**
	40	96.8 (±0.0)	97.8 (±0.0)	**97.8 (±0.2)**	**97.8 (±0.1)**	**97.8 (±0.0)**
	30	97.5 (±0.1)	98.2 (±0.2)	**98.6 (±0.0)**	98.2 (±0.2)	**98.9 (±0.0)**
	20	98.1 (±0.0)	98.4 (±0.1)	**98.9 (±0.2)**	98.6 (±0.0)	**99.0 (±0.0)**
	10	98.2 (±0.1)	98.7 (±0.1)	**99.5 (±0.1)**	98.5 (±0.1)	**99.5 (±0.0)**

Table 3: **Selective accuracy across coverage levels.** SPTD-based methods outperform current SOTA error rates across datasets with full-coverage accuracy alignment. Results show mean and standard deviation over 5 runs. **Bold** marks the best result at each coverage level; underline indicates the best among single-run methods.

favorable coverage/error trade-offs between 25 and 50 checkpoints when considering the full coverage spectrum. However, when considering the high coverage regime in particular (which is what most selective prediction works focus on), even sub-sampling 10 intermediate models is sufficient

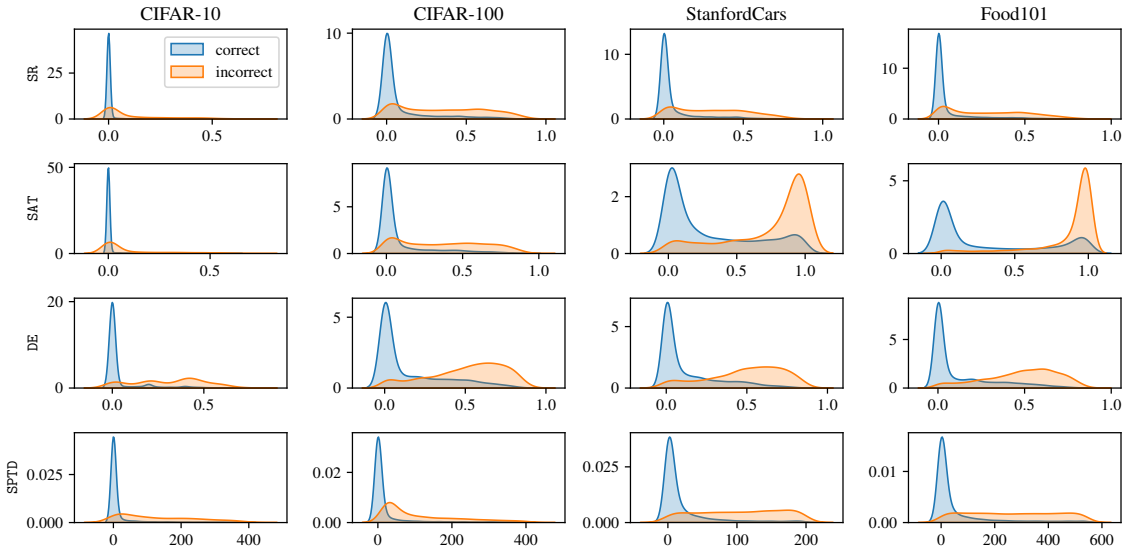


Figure 7: **Distribution of g for different datasets and selective classification methods.** Since all methods are designed to address the selective prediction problem, they all manage to separate correct from incorrect points (albeit at varying success rates). We see that `SPTD` spreads the scores for incorrect points over a wide range with little overlap. We observe that for `SR`, incorrect and correct points both have their mode at approximately the same location which hinders performative selective classification. Although `SAT` and `DE` show larger bumps at larger score ranges, the separation with correct points is weaker as correct points also result in higher scores more often than for `SPTD`.

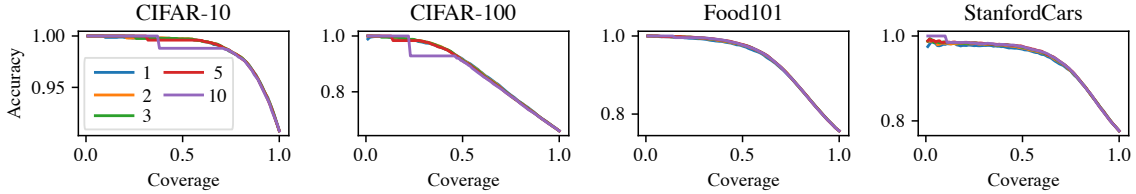


Figure 8: **Coverage/error trade-off of `SPTD` for varying checkpoint weighting k as used in v_t.** We observe strong performance for $k \in [1, 3]$ across datasets.

for SOTA selective classification. Hence, with only observing the training stage, our method's computational overhead reduces to only 10 forward passes at test time when the goal is to reject at most $30\% - 50\%$ of incoming data points. In contrast, `DE` requires to first train E models (with $E = 10$ being a typical and also our particular choice for `DE`) and perform inference on these E models at test time. Further increasing the checkpointing resolution does offer increasingly diminishing returns but also leads to improved accuracy-coverage trade-offs, especially at low coverage.

Detection of Out-of-Distribution and Adversarial Examples. Out-of-distribution (OOD) and adversarial example detection are important disciplines in trustworthy ML related to selective prediction. We therefore provide preliminary evidence in Figure 10 that our method can be used for detecting OOD and adversarial examples. While these results are encouraging, we remark that

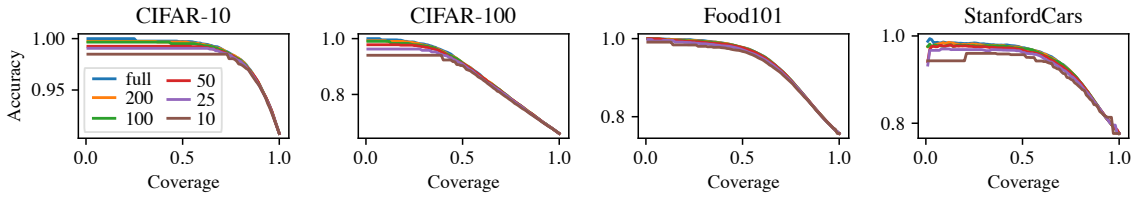


Figure 9: **Coverage/error trade-off of SPTD for varying checkpoint counts**. SPTD delivers consistent performance independent of the checkpointing resolution at high coverage. At low coverage, a more detailed characterization of training dynamics helps.

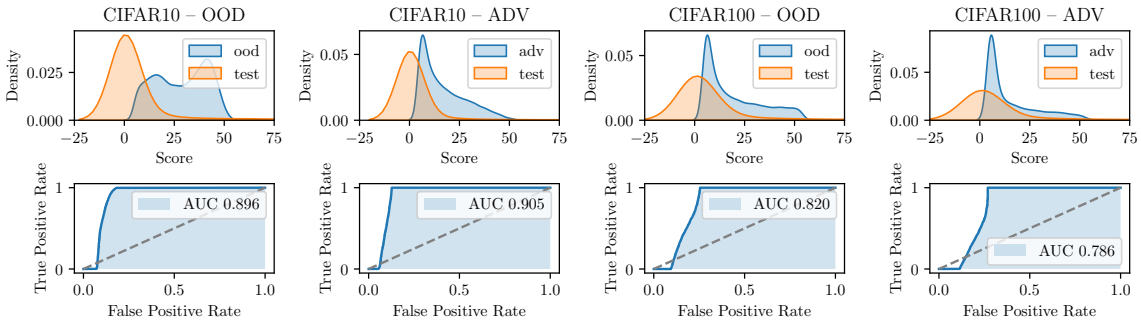


Figure 10: **Performance of SPTD on out-of-distribution (OOD) and adversarial sample detection**. The first row shows the score distribution of the in-distribution CIFAR-10/100 test set vs the SVHN OOD test set or a set consisting of adversarial samples generated via a PGD attack in the final model. The second row shows the effectiveness of a thresholding mechanism by computing the area under the ROC curve. Our score enables separation of anomalous data points from in-distribution test points.

adversarial and OOD samples are less well defined as incorrect data points and can come in a variety of different flavors (i.e., various kinds of attacks or various degrees of OOD-ness). As such, we strongly believe that future work is needed to determine whether a training-dynamics-based approach to selective prediction can be reliably used for OOD and adversarial sample identification.

Cost vs Performance Tradeoff. In Table 4, we report both the time and space complexities for all SC methods at training and test time along with their selective classification performance as per our results in Table 1 and Figure 9. We denote with E the number of DE models and with T the number of SPTD checkpoints. Although SR and SAT are the cheapest methods to run, they also perform the poorest at SC. SPTD is significantly cheaper to train than DE and achieves competitive performance at $T \approx E$. Although DE+SPTD is the most expensive model, it also provides the strongest performance.

E.2.2. EXTENDED SYNTHETIC EXPERIMENTS

We extend the experiment from Figure 6 to all tested SC methods in Figure 11. We also provide an extended result using Bayesian Linear Regression in Figure 12.

Method	Train Time	Train Space	Inf Time	Inf Space	Rank
SR	$O(1)$	$O(1)$	$O(1)$	$O(1)$	5
SAT	$O(1)$	$O(1)$	$O(1)$	$O(1)$	4
DE	$O(E)$	$O(E)$	$O(E)$	$O(E)$	=2
SPTD	$O(1)$	$O(T)$	$O(T)$	$O(T)$	=2
DE+SPTD	$O(E)$	$O(ET)$	$O(ET)$	$O(ET)$	1

Table 4: **Cost vs performance tradeoff in terms of training time/space, inference time/space and the performance rank.** SPTD is comparable in performance (at $T \approx E$) and cheaper to train than DE. DE+SPTD is the most expensive model, but delivers the best performance across datasets.

Table 5: **Selective accuracy achieved across coverage levels for CIFAR-100 with ResNet-50**.

Cov.	SR	SAT+ER+SR	DE	SPTD	DE+SPTD
100	**77.0 (±0.0)**	**77.0 (±0.0)**	**77.0 (±0.0)**	**77.0 (±0.0)**	**77.0 (±0.0)**
90	79.2 (± 0.1)	79.9 (± 0.1)	81.2 (± 0.0)	81.4 (± 0.1)	**82.1 (± 0.1)**
80	83.1 (± 0.0)	83.9 (± 0.0)	85.7 (± 0.1)	85.6 (± 0.1)	**86.0 (± 0.2)**
70	87.4 (± 0.1)	88.2 (± 0.1)	89.6 (± 0.1)	**89.7 (± 0.0)**	**89.8 (± 0.1)**
60	90.5 (± 0.0)	90.8 (± 0.2)	**90.7 (± 0.2)**	90.6 (± 0.0)	**90.9 (± 0.1)**
50	93.4 (± 0.1)	93.8 (± 0.0)	95.3 (± 0.0)	95.1 (± 0.0)	**95.4 (± 0.0)**
40	95.4 (± 0.0)	95.5 (± 0.1)	**97.1 (± 0.1)**	**97.2 (± 0.1)**	**97.2 (± 0.0)**
30	97.4 (± 0.2)	97.7 (± 0.0)	**98.6 (± 0.1)**	**98.6 (± 0.1)**	**98.7 (± 0.0)**
20	97.9 (± 0.1)	98.4 (± 0.1)	99.0 (± 0.0)	99.2 (± 0.1)	**99.2 (± 0.1)**
10	98.1 (± 0.0)	98.8 (± 0.1)	99.2 (± 0.1)	**99.4 (± 0.1)**	**99.6 (± 0.1)**

E.2.3. CIFAR-100 RESULTS WITH RESNET-50

We further provide a full set of results using the larger ResNet-50 architecture on CIFAR-100 in Figure 5.

E.2.4. APPLYING SPTD ON TOP OF SAT

Our main set of results suggest that applying SPTD on top of DE further improves performance. The same effect holds when applying SPTD on top of non-ensemble-based methods such as SAT. We document this result in Figure 13.

E.2.5. ABLATION ON k

We provide a comprehensive ablation on the weighting parameter k in Figures 8 and 14.

E.2.6. COMPARISON WITH LOGIT-VARIANCE APPROACHES

We showcase the effectiveness of SPTD against LOGITVAR [43], an approach that also computes predictions of intermediate models but instead computes the variance of the correct prediction. We adapt this method to our selective prediction approach (for which true labels are not available) by computing the variance over the maximum predicted logit instead of the true logit. In Figure 15,

we see that the weighting of intermediate checkpoints introduced by SPTD leads to stronger performance over the LOGITVAR baseline approach.

E.2.7. ESTIMATING τ ON VALIDATION VS TEST DATA

Consistent with prior works [12, 16, 27, 34], we estimate τ directly on the test set. However, a realistically deployable approach has to compute thresholds based on a validation set for which labels are available. In the case of selective classification, the training, validation, and test sets follow the i.i.d. assumption, which means that an approach that sets the threshold based on a validation set should work performantly on a test set, too. Under consistent distributional assumptions, estimating thresholds on a validation set functions as an unbiased estimator of accuracy/coverage tradeoffs on the test set. By the same virtue, setting thresholds directly on the test set and observing the SC performance on that test set should be indicative for additional test samples beyond the actual provided test set. It is important to remark that the validation set should only be used for setting the thresholds and not for model selection / early stopping which would indeed cause a potential divergence between SC performance on the validation and test sets. Further note that violations of the i.i.d assumption can lead to degraded performance due to mismatches in attainable coverage as explored in [5].

To confirm this intuition, we present an experiment in Figure 16 and Figure 17 where we select 50% of the samples from the test set as our validation set (and maintain the other 50% of samples as our new test set). We first generate 5 distinct such validation-test splits, set the thresholds for τ based on the validation set, and then evaluate selective classification performance on the test set by using the thresholds derived from the validation set. We compare these results with our main approach which sets the thresholds based on the test set directly (ignoring the validation set). We provide an additional experiment where we partition the validation set from the training set in Figure 18. We see that the results are statistically indistinguishable from each other, confirming that this evaluation practice is valid for the selective classification setup we consider.

E.2.8. COMPARING s_{MAX} AND s_{SUM}

As per our theoretical framework and intuition provided in Section 3, the sum score s_{SUM} should offer the most competitive selective classification performance. We confirm this finding in Figure 19 where we plot the accuracy/coverage curves across all datasets for both s_{MAX} and s_{SUM}. Overall, we find that the sum score s_{SUM} consistently outperforms the more noisy maximum score s_{MAX}.

E.3. The Importance of Accuracy Alignment

Our results in Table 1 rely on accuracy alignment: We explicitly make sure to compare all methods on an equal footing by disentangling selective prediction performance from gains in overall utility. This is done by early stopping model training when the accuracy of the worst performing model is reached.

We expand on the important point that many previous approaches conflate both (i) generalization performance and (ii) selective prediction performance into a single score: the area under the accuracy/coverage curve. This metric can be maximized either by improving generalization performance (choosing different architectures or model classes) or by actually improving the ranking of points for selective prediction (accepting correct points first and incorrect ones last). As raised by a variety of recent works [8, 18, 38], it is impossible and problematic to truly assess whether

a method performs better at selective prediction (i.e., determining the correct acceptance ordering) without normalizing for this inherent difference yielded as a side effect by various SC methods. In other words, an SC method with lower base accuracy (smaller correct set) can still outperform another SC method with higher accuracy (larger correct set) in terms of the selective acceptance ordering (an example of which is given in Table 3 of [34]). Accuracy normalization allows us to eliminate these confounding effects between full-coverage utility and selective prediction performance by identifying which models are better at ranking correct points first and incorrect ones last. This is of particular importance when comparing selective prediction methods which change the training pipeline in different ways, as is done in the methods presented in Table 1.

However, when just comparing SPTD to one other method, we do not need to worry about accuracy normalization. Showcasing this, we run SPTD on top of an unnormalized SAT+ER+SR run and provide these experiments in Figure 13. We see that the application of SPTD on top of SAT+ER+SR allows us to further boost performance (similar to the results where we apply SPTD on top of DE in Table 1). So to conclude, experimentally, when using the best model, we see that SPTD still performs better at selective prediction than the relevant baseline for that training pipeline. We wish to reiterate that this issue of accuracy normalization highlights another merit of SPTD, which is that it can easily be applied on top of any training pipeline (including those that lead to the best model) and allows easy comparison to the selective classification method that training pipeline was intended to be deployed with.

E.4. Evaluation using other performance metrics

We further provide results of summary performance metrics across datasets in Table 6:

- The area under the accuracy-coverage curve (AUACC) as discussed in Geifman et al. [18].
- The area under the receiver operating characteristic (AUROC) as suggested by Galil et al. [14].
- The accuracy normalized selective classification score (ANSC) from Geifman et al. [18] and Rabanser et al. [38].

E.5. Evaluation of more competing approaches

We further compare our method with two more contemporary selective prediction approaches:

- AUCOC [41]: This work uses a custom cost function for multi-class classification that accounts for the trade-off between a neural network's accuracy and the amount of data that requires manual inspection from a domain expert.
- CCL-SC [47]: This work proposes optimizing feature layers to reduce intra-class variance via contrastive learning.

Across both methods, we find that they do not outperform ensemble-based methods like DE and hence also do not outperform SPTD. See Table 7 for detailed results.

Table 6: **Evaluation of SC approaches using various evaluation metrics**.

Dataset	Method	$1 - \texttt{AUACC}$	ANSC	AUROC
CIFAR10	SR	0.053 ± 0.002	0.007 ± 0.000	0.918 ± 0.002
	SPTD	0.048 ± 0.001	0.004 ± 0.000	0.938 ± 0.002
	DE	0.046 ± 0.002	0.004 ± 0.000	0.939 ± 0.003
	SAT	0.054 ± 0.002	0.006 ± 0.000	0.924 ± 0.005
	DG	0.054 ± 0.001	0.006 ± 0.000	0.922 ± 0.005
CIFAR100	SR	0.181 ± 0.001	0.041 ± 0.001	0.865 ± 0.003
	SPTD	0.174 ± 0.002	0.037 ± 0.000	0.872 ± 0.002
	DE	0.159 ± 0.001	0.030 ± 0.001	0.880 ± 0.003
	SAT	0.180 ± 0.001	0.041 ± 0.001	0.866 ± 0.003
	DG	0.182 ± 0.001	0.041 ± 0.001	0.867 ± 0.002
GTSRB	SR	0.020 ± 0.002	0.001 ± 0.000	0.986 ± 0.003
	SPTD	0.019 ± 0.002	0.001 ± 0.000	0.986 ± 0.005
	DE	0.015 ± 0.001	0.001 ± 0.000	0.986 ± 0.002
	SAT	0.027 ± 0.001	0.001 ± 0.000	0.984 ± 0.002
	DG	0.019 ± 0.003	0.001 ± 0.000	0.986 ± 0.002
SVHN	SR	0.027 ± 0.000	0.006 ± 0.001	0.895 ± 0.004
	SPTD	0.025 ± 0.003	0.003 ± 0.001	0.932 ± 0.005
	DE	0.021 ± 0.001	0.005 ± 0.000	0.912 ± 0.003
	SAT	0.028 ± 0.001	0.006 ± 0.000	0.895 ± 0.002
	DG	0.026 ± 0.001	0.007 ± 0.000	0.896 ± 0.006

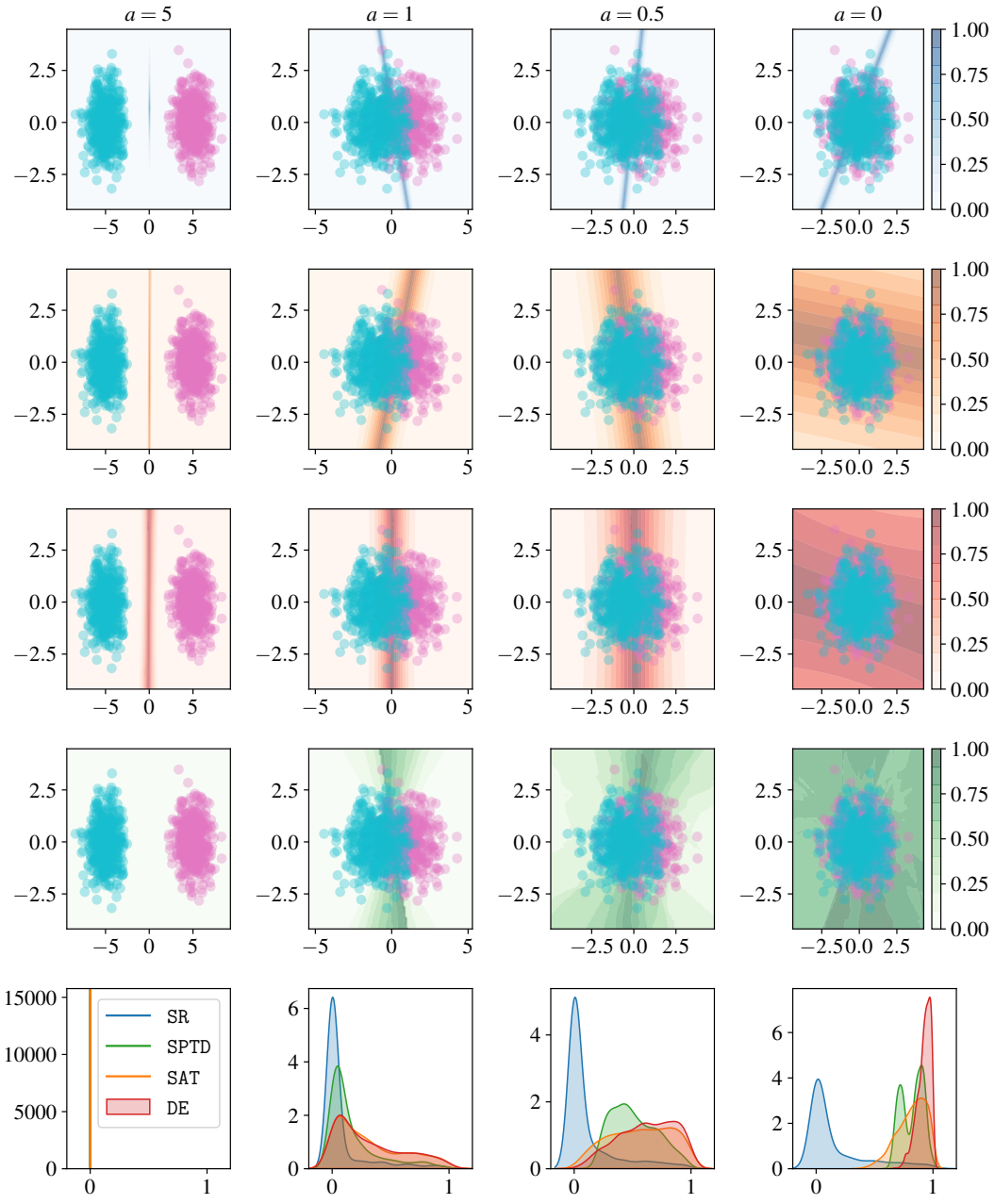


Figure 11: **Extended Gaussian experiment.** The first row corresponds to the anomaly scoring result of SR, the second to the result of SAT, the third to the result of DE, and the fourth to the result of SPTD. The bottom row shows the score distribution for each method over the data points. We see that all methods reliably improve over the SR baseline. At the same time, we notice that SAT and DE still assign higher confidence away from the data due to limited use of decision boundary oscillations. SPTD addresses this limitation and assigns more uniform uncertainty over the full data space.

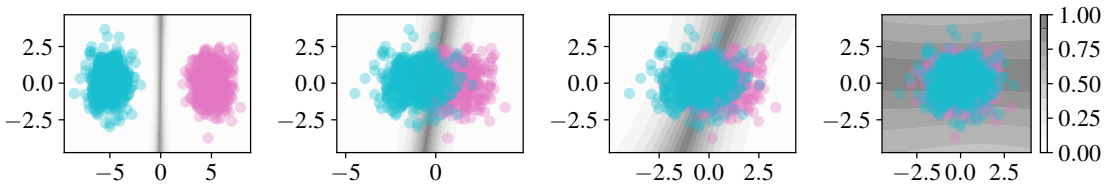


Figure 12: **Bayesian linear regression experiment on Gaussian data.** Results comparable to DE.

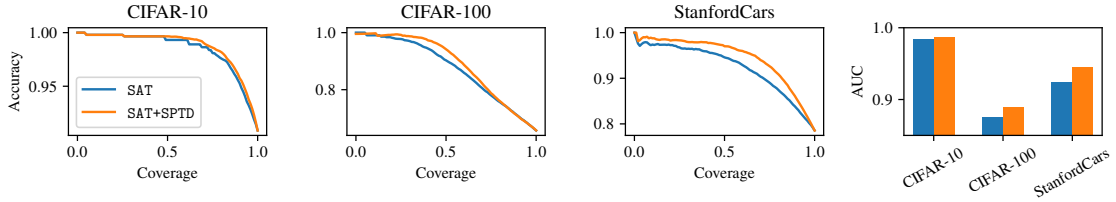


Figure 13: **Applying SPTD on top of SAT.** Similar as with DE, we observe that the application of SPTD improves performance.

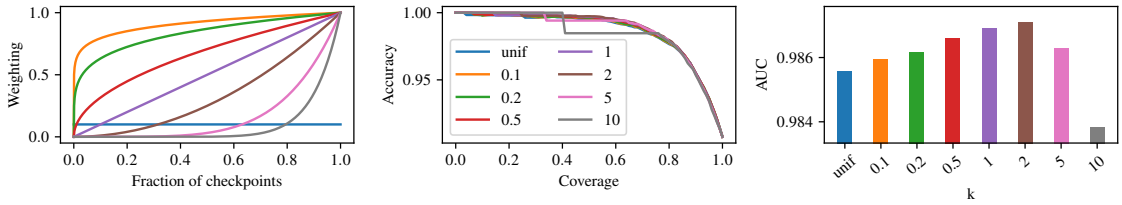


Figure 14: **Extended ablation results on k on CIFAR-10.** We now also consider $k \in (0, 1]$ as well as a uniform weighting assigning the same weight to all checkpoints. We confirm that a convex weighting yields best performance.

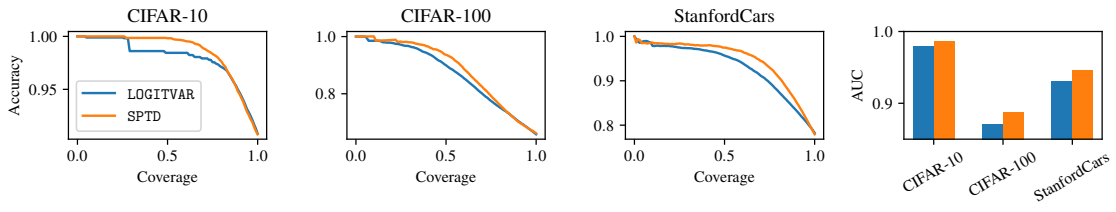


Figure 15: **Comparison of LOGITVAR vs SPTD.** We observe that SPTD, which incorporates weighting of intermediate checkpoints using v_t, outperforms LOGITVAR.

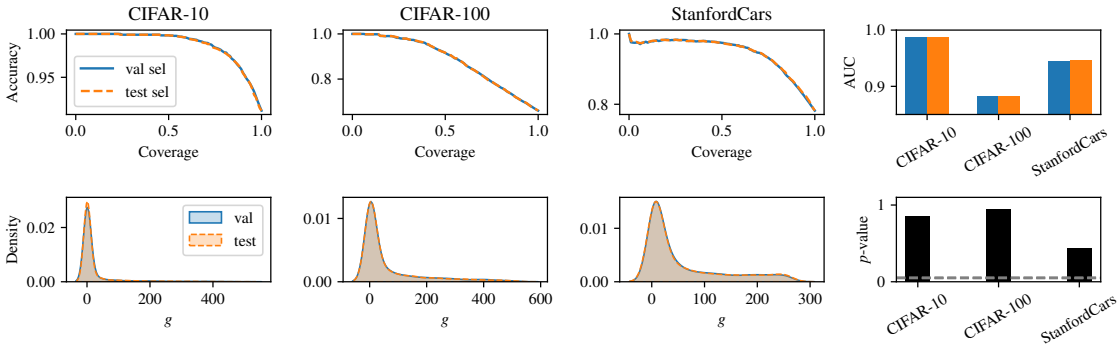


Figure 16: **`SPTD` accuracy/coverage trade-offs and score distributions on test data obtained by computing τ on a validation set or directly on the test set.** The first row shows the obtained accuracy/coverage trade-offs with the respective AUC scores. In the second row, we show the score distribution for both the picked validation and test sets, along with p-values from a KS-test to determine the statistical closeness of the distributions. Overall, we observe that both methods are statistically indistinguishable from each other.

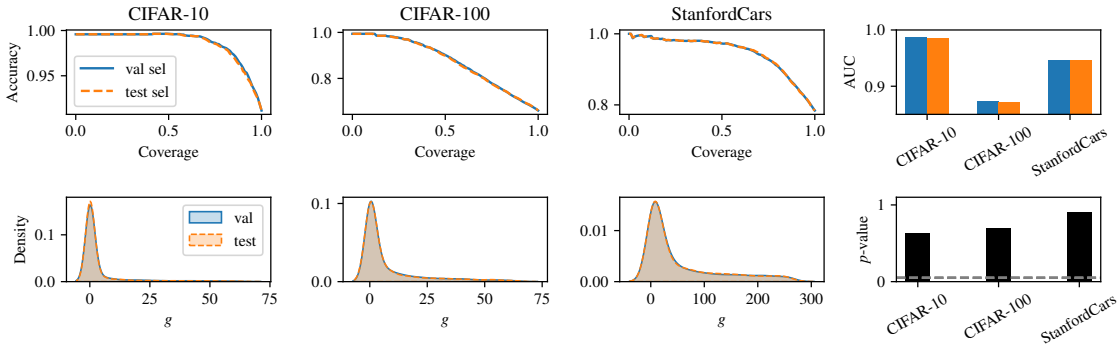


Figure 17: **`SAT` accuracy/coverage trade-offs and score distributions on test data obtained by computing τ on a validation set or directly on the test set.** Same as Figure 16 but with `SAT`.

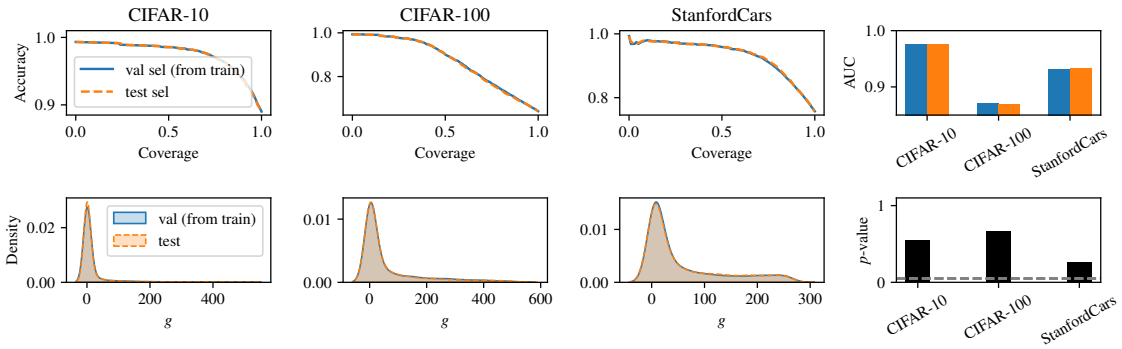


Figure 18: **`SPTD` accuracy/coverage trade-offs and score distributions on test data obtained by computing τ on a validation set or directly on the test set.** Same as Figure 16 but with the validation set is taken from the original training set.

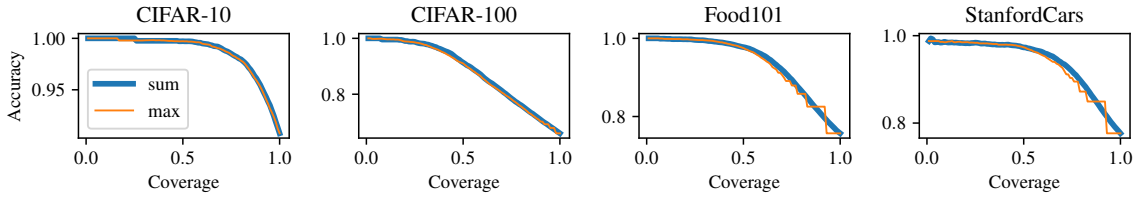


Figure 19: **Comparing s_{MAX} and s_{SUM} performance**. It is clear that s_{SUM} effectively denoises s_{MAX}.

(*a*) CIFAR-10 (*b*) CIFAR-100 (*c*) Food101 (*d*) StanfordCars

Figure 20: **Additional individual examples across datasets.**

Table 7: **Selective accuracy achieved across coverage levels for `AUCOC` and `CCL−SC`**. Similar as Table 1. Neither `AUCOC` nor `CCL−SC` is able to outperform `DE` or `SPTD`. **Bold** numbers are best results at a given coverage level across all methods and underlined numbers are best results for methods relying on a single training run only.

	Coverage	AUCOC	CCL-SC	SPTD	DE
CIFAR-10	100	**92.9** (±0.0)	**92.9** (±0.0)	**92.9** (±0.0)	**92.9** (±0.0)
	90	96.0 (±0.1)	95.9 (±0.2)	96.5 (±0.0)	**96.8** (±0.1)
	80	98.1 (±0.2)	98.0 (±0.3)	98.4 (±0.1)	**98.7** (±0.0)
	70	99.0 (±0.3)	98.5 (±0.2)	99.2 (±0.1)	**99.4** (±0.1)
	60	99.3 (±0.1)	99.1 (±0.2)	**99.6** (±0.2)	**99.6** (±0.1)
	50	99.4 (±0.2)	99.0 (±0.3)	**99.8** (±0.0)	99.7 (±0.0)
	40	99.5 (±0.1)	99.4 (±0.2)	**99.8** (±0.1)	**99.8** (±0.0)
	30	99.5 (±0.2)	99.2 (±0.3)	**99.8** (±0.1)	**99.8** (±0.0)
	20	99.6 (±0.1)	99.4 (±0.2)	**100.0** (±0.0)	99.8 (±0.0)
	10	99.7 (±0.0)	99.4 (±0.1)	**100.0** (±0.0)	99.8 (±0.0)
CIFAR-100	100	**75.1** (±0.0)	**75.1** (±0.0)	**75.1** (±0.0)	**75.1** (±0.0)
	90	78.7 (±0.2)	76.5 (±0.3)	**80.4** (±0.0)	80.2 (±0.0)
	80	83.2 (±0.1)	82.2 (±0.2)	**84.6** (±0.1)	**84.7** (±0.1)
	70	87.4 (±0.1)	86.1 (±0.2)	**88.7** (±0.0)	88.6 (±0.1)
	60	89.8 (±0.2)	88.6 (±0.3)	90.1 (±0.0)	**90.2** (±0.2)
	50	93.3 (±0.1)	92.1 (±0.2)	94.6 (±0.0)	**94.8** (±0.0)
	40	95.9 (±0.2)	95.2 (±0.3)	**96.9** (±0.1)	96.8 (±0.1)
	30	98.2 (±0.1)	96.6 (±0.2)	**98.4** (±0.1)	98.4 (±0.1)
	20	98.6 (±0.2)	98.4 (±0.3)	98.8 (±0.2)	**99.0** (±0.0)
	10	98.8 (±0.1)	98.7 (±0.2)	**99.4** (±0.1)	99.2 (±0.1)