PAC CONFIDENCE SETS FOR DEEP NEURAL NETWORKS VIA CALIBRATED PREDICTION

Sangdon Park
University of Pennsylvania
sangdonp@cis.upenn.edu

Osbert Bastani
University of Pennsylvania
obastani@seas.upenn.edu

Nikolai Matni
University of Pennsylvania
nmatni@seas.upenn.edu

Insup Lee
University of Pennsylvania
lee@cis.upenn.edu

ABSTRACT

We propose an algorithm combining calibrated prediction and generalization bounds from learning theory to construct confidence sets for deep neural networks with PAC guarantees—i.e., the confidence set for a given input contains the true label with high probability. We demonstrate how our approach can be used to construct PAC confidence sets on ResNet for ImageNet, a visual object tracking model, and a dynamics model for the half-cheetah reinforcement learning problem. [1]

1 INTRODUCTION

A key challenge facing deep neural networks is that they do not produce reliable confidence estimates, which are important for applications such as safe reinforcement learning (Berkenkamp et al., 2017), guided exploration (Malik et al., 2019), and active learning (Gal et al., 2017).

We consider the setting where the test data follows the same distribution as the training data (i.e., we do *not* consider adversarial examples designed to fool the network (Szegedy et al., 2014)); even in this setting, confidence estimates produced by deep neural networks are notoriously unreliable (Guo et al., 2017). One intuition for this shortcoming is that unlike traditional supervised learning algorithms, deep learning models typically overfit the training data (Zhang et al., 2017). As a consequence, the confidence estimates of deep neural networks are flawed even for test data from the training distribution since, by construction, they overestimate the likelihood of the training data.

A promising approach to addressing this challenge is *temperature scaling* (Platt, 1999). This approach takes as input a trained neural network $f_{\hat{\phi}}(y \mid x)$—i.e., whose parameters $\hat{\phi}$ have already been fit to a training dataset Z_{train}—which produces unreliable probabilities $f_{\hat{\phi}}(y \mid x)$. Then, this approach rescales these confidence estimates based on a validation dataset to improve their "calibration". More precisely, this approach fits confidence estimates of the form

$$f_{\hat{\phi},\tau}(y \mid x) \propto \exp(\tau \log f_{\hat{\phi}}(y \mid x)),$$

where $\tau \in \mathbb{R}_{>0}$ is a *temperature scaling* parameter that is fit based on the validation dataset. The goal is to choose τ to minimize *calibration error*, which roughly speaking measures the degree to which the reported error rate differs from the actual error rate.

The key insight is that in the temperature scaling approach, only a single parameter τ is fit to the validation data—thus, unlike fitting the original neural network, the temperature scaling algorithm comes with generalization guarantees based on traditional statistical learning theory.

Despite the improved generalization guarantees, these confidence estimates still do not come with theoretical guarantees. We are interested in producing *confidence sets* that satisfy statistical guarantees while being as small as possible. Given a test input $x \in \mathcal{X}$, a confidence set $C_T(x) \subseteq \mathcal{Y}$

[1] Our code is available at https://github.com/sangdon/PAC-confidence-set.

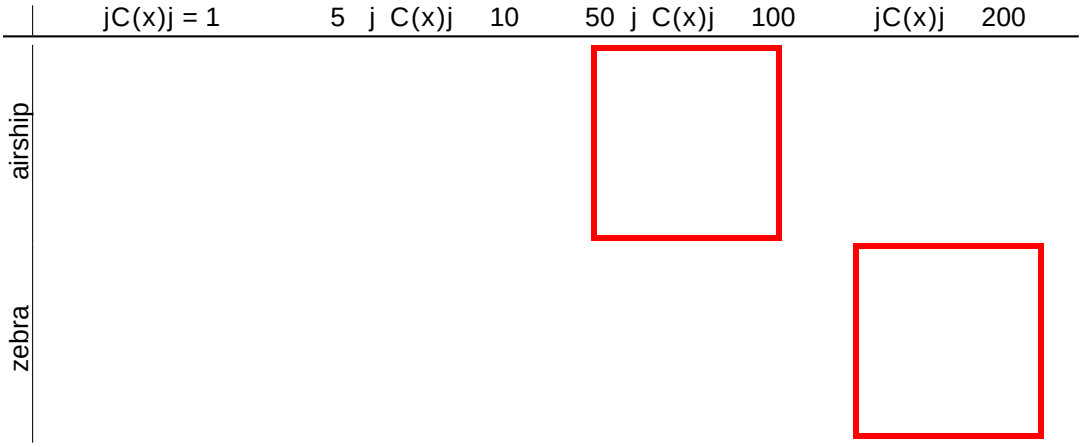


Table 1: ImageNet images with varying ResNet con dence set sizes. The con dence set sizes are on the top. The true label is on the left-hand side. Incorrectly labeled images are boxed in red.

(parameterized by $T \in \mathbb{R}$) should contain the true label y for at least a $1-\alpha$ fraction of cases:

$$P_{(x,y)\sim D}[y \in C_T(x)] \geq 1-\alpha.$$

Since we are tting a parameter T to based on Z_{val}, we additionally incur a probability of failure due to the randomness in Z_{val}. In other words, given $\alpha, \delta \in \mathbb{R}_{>0}$, we aim to obtain probably approximately correct (PAC) con dence sets $C_T(x)$ satisfying the guarantee

$$P_{Z_{val}\sim D^n}\left[P_{(x,y)\sim D}(y \in C_T(x)) \geq 1-\alpha\right] \geq 1-\delta.$$

Indeed, techniques from statistical learning theory (Vapnik, 1999) can be used to do so (Vovk, 2013).

There are a number of reasons why con dence sets can be useful. First, they can be used to inform safety critical decision making. For example, consider a doctor who uses prediction tools to help perform diagnosis. Having a con dence set would both help the doctor estimate the con dence of the prediction (i.e., smaller con dence sets imply higher con dence), but also give a sense of the set of possible diagnoses. Second, having a con dence set can be useful for reasoning about safety since they contain the true outcome with high probability. For instance, robots may use a con dence set over predicted trajectories to determine whether it is safe to act with high probability. As a concrete example, consider a self-driving car that uses a deep neural network to predict the path that a pedestrian might take. We require that the self-driving car avoid the pedestrian with high probability, which it can do by avoiding all possible paths in the predicted con dence set.

Contributions. We propose an algorithm combining calibrated prediction and statistical learning theory to construct PAC con dence sets for deep neural networks (Section 3). We propose instantiations of this framework in the settings of classi cation, regression, and learning models for reinforcement learning (Section 3.6). Finally, we evaluate our approach on three benchmarks: ResNet (He et al., 2016) for ImageNet (Russakovsky et al., 2015), a model (Held et al., 2016) learned for a visual object tracking benchmark (Wu et al., 2013), and a probabilistic dynamics model (Chua et al., 2018) learned for the half-cheetah environment (Brockman et al., 2016) (Section 4). Examples of ImageNet images with different sized ResNet con dence sets are shown in Table 1. As can be seen, our con dence sets become larger and the images become more challenging to classify. In addition, we show predicted con dence sets for ResNet in Table 2, as well as predicted con dence sets for the visual object tracking model in Table 3.

Related work. There has been work on constructing con dence sets with theoretical guarantees. Oftentimes, these guarantees are asymptotic rather than nite sample (Steinberger & Leeb, 2016; 2018). Alternatively, there has been work focused on predicting con dence sets with a given expected size (Denis & Hebiri, 2017).

More relatedly, there has been recent work on obtaining PAC guarantees. For example, there has been some work speci c prediction tasks such as binary classi cation (Lei, 2014; Wang & Qiao,

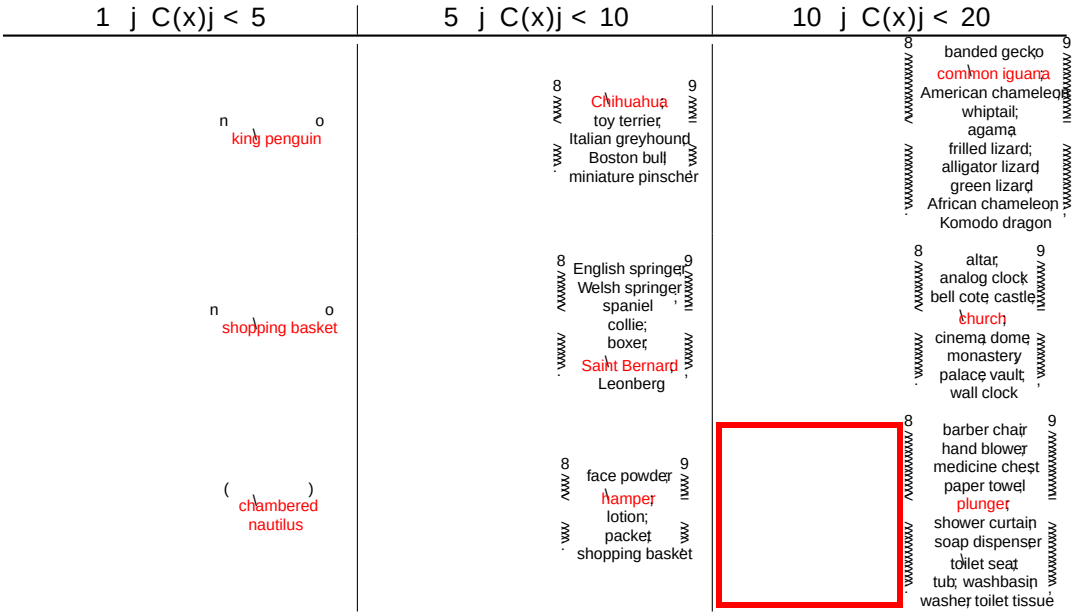


Table 2: Con dence sets of ImageNet images with varying ResNet con dence set sizes. The predicted con dence set is shown to the right of the corresponding input image. The true label is shown in red, and the predicted label is shown with a hat. See Table 5 in Appendix D for more examples.

Table 3: Visualization of con dence sets for the tracking dataset (Wu et al., 2013), including the ground truth bounding box (white), the bounding box predicted by the original neural network (Held et al., 2016) (red), and the bounding box produced using our con dence set predictor (green). We have overapproximated the predicted ellipsoid con dence set with a box. Our bounding box contains the ground truth bounding box with high probability. See Table 9 in Appendix D for more examples.

2018). There has also been work in the setting of regression (Lei et al., 2018; Barber et al., 2019). However, in this case, the con dence sets are xed in size—i.e., they do not depend on the input x (Barber et al., 2019). Furthermore, they make stability assumptions about the learning algorithm (though they achieved improved rates by doing so) (Lei et al., 2018; Barber et al., 2019).

The most closely related work is on conformal prediction (Papadopoulos, 2008; Vovk, 2013). Like our approach, this line of work provides a way to construct con dence sets from a given con dence predictor, and provided PAC guarantees for the validity of these con dence sets. Indeed, with some work, our generalization bound Theorem 1 can be shown to be equivalent to Theorem 1 in Vovk (2013). In contrast to their approach, we proposed to use calibrated prediction to construct con - dence predictors that can suitably be used with deep neural networks. Furthermore, our approach

makes explicit the connections to temperature scaling and as well as to generalization bounds from statistical learning theory (Vapnik, 1999). In addition, unlike our paper, they do not explicitly provide an efficient algorithm for constructing confidence sets. Finally, we also propose an extension to the case of learning models for reinforcement learning.

Finally, we build on a long line of work on *calibrated prediction*, which aims to construct "calibrated" probabilities (Murphy, 1972; DeGroot & Fienberg, 1983; Platt, 1999; Zadrozny & Elkan, 2001; 2002; Naeini et al., 2015; Kuleshov & Liang, 2015). Roughly speaking, probabilities are calibrated if events happen at rates equal to the predicted probabilities. This work has recently been applied to obtaining confidence estimates for deep neural networks (Guo et al., 2017; Kuleshov et al., 2018; Pearce et al., 2018), including for learned models for reinforcement learning (Malik et al., 2019). However, these approaches do not come with PAC guarantees.

2 PAC CONFIDENCE SETS

Our goal is to estimate confidence sets that are as small as possible, while simultaneously ensuring that they are *probably approximately correct* (PAC) (Valiant, 1984). Essentially, a confidence set is correct if it contains the true label. More precisely, let X be the inputs and Y be the labels, and let D be a distribution over $Z = X \times Y$. A confidence set predictor is a function $C : X \to 2^Y$ such that $C(x) \subseteq Y$ is a set of labels; we denote the set of all confidence set predictors by $\mathcal{C}$. For a given example $(x, y) \sim D$, we say C is correct if $y \in C(x)$. Then, the error of C is

$$L(C) = P_{(x,y) \sim D}[y \notin C(x)]. \tag{1}$$

Finally, consider an algorithm A that takes as input a validation set $Z_{val} \in Z^*$ consisting of n i.i.d. samples $(x, y) \sim D$, and outputs a confidence set predictor $\hat{C}$. Given $\epsilon, \delta \in \mathbb{R}_{>0}$, we say that A is probably approximately correct (PAC) if

$$P_{Z_{val} \sim D^n}\left[L(\hat{C}) > \epsilon \text{ where } \hat{C} = A(Z_{val}) \right] < \delta. \tag{2}$$

Our goal is to design an algorithm A that satisfies (2) while constructing confidence sets $C(x)$ that are as "small in size" as possible on average. The size of $C(x)$ depends on the domain. For classification, we consider confidence sets that are arbitrary subsets of labels, $Y = \{1, \ldots, Y\}$, and we measure the size by $|C(x)| \in \mathbb{N}$—i.e., the number of labels in $C(x)$. For regression, we consider confidence sets that are intervals, $C(x) = [a, b] \subseteq Y = \mathbb{R}$, and we measure size by $b - a$—i.e., the length of the predicted interval. Note that there is an intrinsic tradeoff between satisfying (2) and average size of $C(x)$—larger confidence sets are more likely to satisfy (2).

3 PAC ALGORITHM FOR CONFIDENCE SET CONSTRUCTION

Our algorithm is formulated in the *empirical risk* framework. Typically, this framework refers to empirical risk minimization. In our setting, such an algorithm would take as input (i) a parametric family of confidence set predictors $\mathcal{C} = \{C_\theta \mid \theta \in \Theta\}$, where Θ is the parameter space, and (ii) a training set $Z_{val} \in Z^*$ of n i.i.d. samples $(x, y) \sim D$, and output the confidence set predictor $C_{\hat{\theta}}$, where $\hat{\theta} \in \Theta$ minimizes the empirical risk:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \hat{L}(C_\theta; Z_{val}) \quad \text{where} \quad \hat{L}(C; Z_{val}) = \frac{1}{n} \sum_{(x,y) \in Z_{val}} I[y \notin C(x)].$$

Here, $I[\cdot] \in \{0, 1\}$ is the indicator function, and the empirical risk $\hat{L}$ is an estimate of the confidence set error (1) based on the validation set Z_{val}.

However, our algorithm does not minimize the empirical risk. Rather, recall that our goal is to minimize the size of the predicted confidence sets given a PAC constraint on the true risk $L(\hat{\theta})$ based on the given PAC parameters $\epsilon, \delta \in \mathbb{R}_{>0}$ and the number of available validation samples $n = |Z_{val}|$. Thus, the risk shows up as a constraint in the optimization problem, and the objective is instead to minimize the size of the predicted confidence sets:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} S(\theta) \quad \text{subj. to } \hat{L}(C_\theta; Z_{val}) \leq \epsilon. \tag{3}$$

At a high level, the value $= (n; ;) 2 R_{0}$ is chosen to enforce the PAC constraint, and is based on generalization bounds from statistical learning theory (Valiant, 1984). Furthermore, following the temperature scaling approach (Platt, 1999), the parameter space is chosen to be as small as possible (in particular, one dimensional) to enable good generalization. Finally, our choice of size metric S follows straightforwardly based on our choice of parameter space. In the remainder of this section, we describe the choices of (i) parameter space, (ii) size metric $S( )$, and (iii) con dence level $(n; ;)$ in more detail, as well as how to solve (3) given these choices.

3.1 Choice of Parameter Space

Probability forecasters. Our construction of the parameteric family of con dence set predictors C assumes given a probability forecaster $f : X ! P_{Y}$, where P_{Y} is a space of probability distributions over Y. Given such an f, we use $f (y j x)$ to denote the probability of label y under distribution $f (x)$. Intuitively, $f (y j x)$ should be the probability (or probability density) that y is the true label for a given input x—i.e., $f (y j x) P_{(X;Y) D}[Y = y j X = x]$. For example, in classi cation, we can choose P_{Y} to be the space of categorical distributions over Y, and f may be a neural network whose last layer is a softmax layer with jYj outputs. Then, $f (y j x) = f (x)_{y}$. Alternatively, in regression, we can choose P_{Y} to be the space of Gaussian distributions, and f may be a neural network whose last layer outputs the values $((x); (x)) 2 R R_{>0}$ of a Gaussian distribution. Then, $f (y j x) = N (x; (x); (x)^{2})$, where $((x); (x)) = f (x)$, and $N (; ; ^{2})$ is the Gaussian density function with mean and variance ^{2}.

Training a probability forecaster. To train a probability forecaster, we use a standard approach to calibrated prediction that combines maximum likelihood estimation with temperature scaling[2]. First, we consider a parametric model family $F = f f j 2 g$, where is the parameter space. Note that can be high-dimensional—e.g., the weights of a neural network model. Given a training set $Z_{train} Z$ of m i.i.d. samples $(x; y) D$, the maximum likelihood estimate (MLE) of is

$$\hat{} = \arg\min_{2} `( ; Z_{train}) \quad \text{where} \quad `(; Z_{train}) = \sum_{(x;y)2Z_{train}} \log f (y j x): \tag{4}$$

We could now use $f_{\hat{}}$ as the probability forecaster. However, the problem with directly using $\hat{}$ is that because $\hat{}$ may be high-dimensional, it often over ts the training data Z_{train}. Thus, the probabilities are typically overcon dent compared to what they should be.

To reduce their con dence, we use the temperature scaling approach to calibrate the predicted probabilities (Platt, 1999; Guo et al., 2017). Intuitively, this approach is to train an MLE estimate using exactly the same approach used to train $\hat{}$, but using a single new parameter $2 R_{>0}$. The key idea is that this time, the model family is based on the parameter $\hat{}$ from (4). In other words, the "shape" of the probabilities forecast by are preserved, but their exact values are shifted.

More precisely, consider the model family $F^{0} = f f_{\hat{};} \quad j \quad 2 R_{>0} g$, where

$$f_{\hat{};} (y j x) / \exp \quad \log f_{\hat{}}(y j x) \quad :$$

Then, we have the following MLE for :

$$\hat{} = \arg\min_{2 R_{>0}} `^{0}( ; Z^{0}_{train}) \quad \text{where} \quad `^{0}(; Z^{0}_{train}) = \sum_{(x;y)2Z^{0}_{train}} \log f_{\hat{};} (y j x): \tag{5}$$

Note that $\hat{}$ is estimated based on a second training set Z^{0}_{train}. Because we are only tting a single parameter, this training set can be much smaller than the training set Z_{train} used to t $\hat{}$.

Parametric family of con dence set predictors. Finally, given a probability forecaster f, we consider one dimensional parameter space $= R$; in an analogy to the temperature scaling technique for calibrated prediction, we denote this parameter by T. In particular, we assume a con dence probability predictor f is given, and consider

$$C_{T}(x) = f y 2 Y j f (y j x) \quad e^{T} g:$$

[2] A priori, it is not obvious that using temperature scaling can improve our con dence set predictor; we give a detailed discussion in Appendix A.1.

In other words, $C_T(x)$ is the set of y with high probability given x according to f. Considering this scalar parameter space, we denote the minimum of (3) by $\hat{T}$.

3.2 Choice of Size Metric $S(T)$

To choose the size metric $S(T)$, we note that for our chosen parametric family of confidence set predictors, smaller values correspond to uniformly smaller confidence sets—i.e.,

$$T \leq T' \Rightarrow \forall x,\ C_T(x) \subseteq C_{T'}(x).$$

Thus, we can simply choose the size metric to be

$$S(T) = T. \tag{6}$$

This choice minimizes the size of the confidence sets produced by our algorithm.

3.3 Choice of Confidence Level $\epsilon(n, \delta, \alpha)$

Naive approach based on VC generalization bound. A naive approach to choosing $\epsilon(n, \delta, \alpha)$ is to do so based on the VC dimension generalization bound (Vapnik, 1999). It is not hard to show that the problem of estimating $\hat{T}$ is equivalent to a binary classification problem, and that the VC dimension of $\mathcal{F}$ for this problem is 1. Thus, the VC dimension bound implies that for all $T \in \mathcal{T}$,

$$P_{Z_{val} \sim D^n}\left(L(C_T) \leq \hat{L}(C_T; Z_{val}) + \sqrt{\frac{\log(2n) + 1 - \log(\delta/4)}{n}}\right) \geq 1 - \delta. \tag{7}$$

The details of this equivalence are given in Appendix B.2. Then, suppose we choose

$$\epsilon(n, \delta, \alpha) = \sqrt{\frac{\log(2n) + 1 - \log(\delta/4)}{n}}.$$

With this choice, for the solution $\hat{T}$ of (3) with $\alpha = \epsilon(n, \delta, \alpha)$, the constraint in (3) ensures that $\hat{L}(C_{\hat{T}}; Z_{val}) \leq \epsilon(n, \delta, \alpha)$. Together with the VC generalization bound (7), we have

$$P_{Z_{val} \sim D^n}\left(L(C_{\hat{T}}) > \alpha\right) < \delta,$$

which is exactly desired the PAC constraint on our predicted confidence sets.

Direct generalization bound. In fact, we can get better choices of ϵ by directly bounding generalization error. For instance, in the realizable setting (i.e., we always have $\hat{L}(C_{\hat{T}}; Z_{val}) = 0$), we can get rates of $n = O(1/\alpha)$ instead of $n = O(1/\alpha^2)$ (Kearns & Vazirani, 1994); see Appendix A.2 for details. We can achieve these rates by choosing $\epsilon = 0$, but then, the PAC guarantees we obtain may actually be stronger than desired (i.e., for $\delta \ll \alpha$). Intuitively, we can directly prove a bound that interpolates between the realizable setting and the VC generalization bound—in particular:

Theorem 1. For any $\alpha \in [0, 1]$, $n \in \mathbb{N}_{>0}$, and $k \in \{0, 1, \ldots, n\}$, we have

$$P_{Z_{val} \sim D^n}\left(L(C_{\hat{T}}) > \alpha\right) \leq \sum_{i=0}^{k} \binom{n}{i} \alpha^i (1-\alpha)^{n-i},$$

where $\hat{T}$ is the solution to (3) with $\alpha = k/n$. [3]

We give a proof in Appendix B.2. Based on Theorem 1, we can choose

$$\epsilon(n, \delta, \alpha) = \max_{k \in \mathbb{N} \cup \{-1\}} k/n \quad \text{subj. to} \quad \sum_{i=0}^{k} \binom{n}{i} \alpha^i (1-\alpha)^{n-i} < \delta. \tag{8}$$

3.4 Theoretical Guarantees

We have the following guarantee, which follows straightforwardly from Theorem 1:

Corollary 1. Let $\hat{T}$ be the solution to (3) for $\alpha = \epsilon(n, \delta, \alpha)$ chosen according to (8). Then, we have

$$P_{Z_{val} \sim D^n}\left(L(C_{\hat{T}}) > \alpha\right) < \delta.$$

In other words, our algorithm is probably approximately correct.

[3] The theorem statement relies on additional standard technical conditions; see Appendix B.1.

Algorithm 1 Algorithm for solving (3).

procedure ESTIMATECONFIDENCESETPREDICTOR(Z_{train}, Z'_{train}, Z_{val})
  Estimate $\hat{\ }$, $\hat{\ }$ using (4) and (5), respectively
  Compute $\ (n; \ ; \ )$ according to (8) by enumerating $k \in \{0, 1, \ldots, n\}$
  Let $k = n \quad (n; \ ; \ )$ (note that $k \in \{0, 1, \ldots, n\}$)
  Sort $(x, y) \in Z_{\text{val}}$ in ascending order of $f_{\hat{\ },\hat{\ }}(y \mid x)$
  Let (x_{k+1}, y_{k+1}) be the $(k+1)$st element in the sorted Z_{val}
  Solve (3) by choosing $\hat{T} = \quad \log f_{\hat{\ },\hat{\ }}(y_{k+1} \mid x_{k+1})$
  Return $C_{\hat{T}} : x \mapsto \{y \in Y \mid f_{\hat{\ },\hat{\ }}(y \mid x) \quad e^{-\hat{T}}\}$
end procedure

3.5 PRACTICAL IMPLEMENTATION

Our algorithm for estimating a confidence set predictor $C_{\hat{T}}$ is summarized in Algorithm 1. The algorithm solves the optimization problem (3) using the choices of $S(T)$, and $(n; \ ; \ )$ described in the preceding sections. There are two key implementation details that we describe here.

Computing $(n; \ ; \ )$. To compute $(n; \ ; \ )$, we need to solve (8). A straightforward approach is to enumerate all possible choices of $k \in \{0, 1, \ldots, n\}$. There are two optimizations. First, the objective is monotone increasing in k, so we can enumerate k in ascending order until the constraint no longer holds. Second, rather than re-compute the left-hand side of the constraint $\sum_{i=0}^{k} \binom{n}{i} \ ^{i}(1 \quad)^{n-i}$, we can accumulate the sum as we iterate over k. We can also incrementally compute $\binom{n}{i}$, ^{i}, and $(1 \quad)^{n-i}$. For numerical stability, we perform these computations in log space.

Solving (3). To solve (3), note that the constraint in (3) is equivalent to

$$\sum_{(x,y) \in Z_{\text{val}}} E(x, y; T) \quad n \quad (n; \ ; \ ) \quad \text{where } E(x, y; T) = \left[f_{\hat{\ },\hat{\ }}(y \mid x) < e^{-T} \right]. \tag{9}$$

Also, note that $k = n \quad (n; \ ; \ )$ is an integer due to the definition of $(n; \ ; \ )$ in (8). Thus, we can interpret (9) as saying that $E(x, y; T) = 1$ for at most k of the points $(x, y) \in Z_{\text{val}}$.

In addition, note that $E(x, y; T)$ decreases monotonically as $f_{\hat{\ },\hat{\ }}(y \mid x)$ becomes larger. Thus, we can sort the points $(x, y) \in Z_{\text{val}}$ in ascending order of $f_{\hat{\ },\hat{\ }}(y \mid x)$, and require that only the first k points (x, y) in this list satisfy $E(x, y; T) = 1$. In particular, letting (x_{k+1}, y_{k+1}) be the $(k+1)$st point, (9) is equivalent to

$$f_{\hat{\ },\hat{\ }}(y_{k+1} \mid x_{k+1}) \quad e^{-T}. \tag{10}$$

In other words, this constraint says that T must satisfy $y_{k+1} \in C_T(x_{k+1})$. Finally, the solution $\hat{T}$ to (3) is the smallest T that satisfies (10), which is the T that makes (10) hold with equality—i.e.,

$$\hat{T} = \quad \log f_{\hat{\ },\hat{\ }}(y_{k+1} \mid x_{k+1}). \tag{11}$$

We have assumed $f_{\hat{\ },\hat{\ }}(y_{k+1} \mid x_{k+1}) > f_{\hat{\ },\hat{\ }}(y_k \mid x_k)$; if not, we decrement k until this holds.

3.6 PROBABILITY FORECASTERS FOR SPECIFIC TASKS

We briefly discuss the architectures we use for probability forecasters for various tasks. We give details, including how we measure the sizes of predicted confidence sets $C_T(x)$, in Appendix C.

We consider three tasks: classification, regression, and model-based reinforcement learning. For classification, we use the standard approach of using a soft-max layer to predict label probabilities $f(y \mid x)$. For regression, we also use a standard approach where the neural network predicts both the mean (x) and covariance (x) of a Gaussian distribution $\mathcal{N}((x); (x))$; then, $f(y \mid x) = \mathcal{N}(y; (x); (x))$ is the probability density of y according to this Gaussian distribution.

Finally, for model-based reinforcement learning, our goal is to construct confidence sets over trajectories predicted using a learned model of the dynamics. We consider unknown dynamics

Figure 1: Results on ResNet for ImageNet with = 20000. Default parameters are = 0:01 and = 10 5. We plot the median and min/max con dence set sizes. (a) Ablation study; "C" is "calibrated predictor" (i.e., use f̂ ^ ^ instead of f ^), and "D" is "direct bound" (i.e., use Theorem 1 instead of the VC generalization bound). (b) Restricted to correctly vs. incorrectly labeled images. (c) Varying . (d) Varying .

g (x^0 j x; u) mapping a state-action pair (x; u) to a distribution over states x^0, and consider a known (and xed) policy (u j x) mapping a given state x to a distribution over actions u 2 U R^{d_U}. Then, we let f (x^0 j x) = E (ujx)[g (x^0 j u)] denote the (unknown) closed-loop dynamics.

Next, we consider a forecaster F(x^0 j x) f (x^0 j x) of the form f (x^0 j x) = N (x^0; (x); (x)), and our goal is to construct con dence sets for the predictions of f ; however, we want to do so for not just for one-step predictions, but for predictions over a time horizon H 2 N. In particular, given initial state x$_0$ 2 X , we can sample x$_{1:H}$ = (x$_1$; :::; x$_H$) f by letting x$_0$ = x$_0$ and sequentially sampling x$_{t+1}$ f (j x$_t$) for each t 2 f 0; 1; :::; H 1g. Then, our goal is to construct a con dence set that contains x$_{1:H}$ 2 X ^{H} with high probability (over both the randomness in an initial state distribution x$_0$ d$_0$ and the randomness in f).

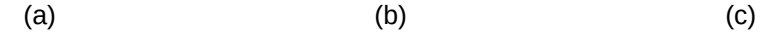
To do so, we construct and use a forecaster F(x$_{1:H}$ j x$_0$) based on f . In principle, this task is a special case of multivariate regression, where the inputs X are (i.e., the initial state x$_0$) and the outputs are Y = X ^{H} (i.e., a predicted trajectory x$_{1:H}$). However, the variance (x) predicted by our probability forecaster is only for a single step, and does not take into account the fact that x$_t$ is itself uncertain. Thus, we use a simple heuristic where we accumulate variances over time. More precisely, we construct (i) the predicted mean x$_{1:H}$ = (x$_1$; :::; x$_H$) by x$_0$ = x$_0$ and x$_{t+1}$ = (x$_t$) for t 2 f 0; 1; :::; H 1g, and (ii) the predicted variances ~$_{1:H}$ = (~$_1$; :::; ~$_H$) by

$$\tilde{}_t = (x_0) + (x_1) + ::: + (x_{t\ 1}):$$

We use a probability forecaster F(x$_{1:H}$ j x$_0$) = N (x$_{1:H}$; x$_{1:H}$; ~$_{1:H}$) to construct con dence sets.

4 EXPERIMENTS

We describe our experiments on ImageNet (a classi cation task), a visual object tracking benchmark (a regression task), and the half-cheetah environment (a model-based reinforcement learning task). We give additional results in Appendix D.



(a) (b) (c)

Figure 2: Con dence set sizes for an object tracking benchmark (Wu et al., 2013); we use $n = 5{,}000$, $\alpha = 0.01$, and $\delta = 10^{-5}$. (a) Ablation study similar to Figure 3. In (b) and (c), we show how the con dence set sizes produced using our algorithm vary with respect to α and δ, respectively.

ResNet for ImageNet. We use our algorithm to compute con dence sets for ResNet (He et al., 2016) on ImageNet (Russakovsky et al., 2015), for $\alpha = 0.01$, $\delta = 10^{-5}$, and $n = 20000$ validation images. We show the results in Figure 1. In (a), we compare our approach to an ablation. In particular, C refers to performing an initial temperature scaling step to calibrate the neural network predictor (i.e., using $\hat{f}^{\wedge}$ instead of $\hat{f}_{\hat{T},\hat{}}^{\wedge}$), and (ii) D refers to using Theorem 1 instead of the VC generalization bound. Thus, C + D refers to our approach. As can be seen, using calibrated predictor produces a noticeable reduction in the maximum con dence set size.

We also compared to the ablation C—i.e., using the VC generalization bound. However, we were unable to obtain valid con dence sets for our choice of α and δ—i.e., (3) is infeasible. That is, using Theorem 1 outperforms using the VC generalization bound since the VC bound is too loose to satisfy the PAC criterion for our choice of parameters. In addition, in Table 6 in Appendix D, we show results for larger choices of α and δ; these results show that our approach substantially outperforms the ablation based on the VC bound even when the VC bound produces valid con dence sets.

In (b), we show the con dence set sizes for images correctly vs. incorrectly labeled by ResNet. As expected, the sizes are substantially larger for incorrectly labeled images. Finally, in (c) and (d), we show how the sizes vary with α and δ, respectively. As expected, the dependence is much more pronounced (note that is log-scale).

Visual object tracking. We apply our con dence set prediction algorithm to a 2D visual single-object tracking task, which is a multivariate regression problem. Speci cally, the input space $\mathcal{X}$ consists of the previous image, the previous bounding box $\mathbb{R}^4$, and the current image. The output space $\mathcal{Y} = \mathbb{R}^4$ is a current bounding box. We use the regression-based tracker from Held et al. (2016), and retrain the regressor neural network to predict the mean and variance of a Gaussian distribution. More precisely, our object tracking model predicts the mean and variance of each bounding box parameter—i.e., $(x_{min}, y_{min}, x_{max}, y_{max})$. Given this bounding box forecaster $\hat{f}$, we calibrate and estimate a con dence set predictor as described in Section 3.6.

We use the visual object tracking benchmark from Wu et al. (2013) to train and evaluate our con-dence set predictor. This benchmark consists of 99 video sequences labeled with ground truth bounding boxes. We randomly split these sequences to form the training set for calibration, validation set for con dence set estimation, and test set for evaluation. For each sequence, a pair of two adjacent frames constitute a single example. Our training dataset contains 20,882 labeled examples, each consisting of of a pair of consecutive images and ground truth bounding boxes. The validation set for con dence set estimation and test set contain 22,761 and 22,761 labeled examples, respectively. Figure 2 shows the sizes of the predicted con dence sets; the sizes are measured as described in Section 3.6 for regression tasks. As for ResNet, we omit results for the VC bound ablation since n is too small to get a bound. The trends are similar to the ones for ResNet.

Half-cheetah. We use our algorithm to compute con dence sets for a probabilistic neural network dynamics model (Chua et al., 2018) for the half-cheetah environment (Brockman et al., 2016), for $\alpha = 0.01$, $\delta = 10^{-5}$, $H = 20$ time steps, and $n = 5000$ validation rollouts. When using temperature scaling to calibrate $\hat{f}^{\wedge}$ to obtain $\hat{f}_{\hat{T},\hat{}}^{\wedge}$, we calibrate each dimension of time steps independently (i.e., we t H parameters, where H is time horizon). We show the results in Figure 3.

Figure 3: Results on the dynamics model for the half-cheetah with $n = 5000$. Default parameters are $\alpha = 0.01$ and $\delta = 10^{-5}$. (a) Ablation study; A is "accumulated variance" (i.e., for each $t \in \{1, \ldots, 20\}$, use $\tilde{\epsilon}_t$ instead of $\epsilon_t = \epsilon(x_{t-1})$), and C and D are as for ResNet. We plot the median and min/max confidence set sizes (see Section 3.6), averaged across $t \in \{1, \ldots, 20\}$. (b) Same ablations, but with per time step size. We plot the average size of the confidence set for the predicted state x_t on step t, as a function of $t \in \{1, \ldots, 20\}$. (c) Varying α, and (d) varying δ.

In (a), we compare to two ablations. The labels C and D are as for ResNet; in addition, A refers to using the accumulated variance $\tilde{\epsilon}_t$ instead of the one-step predicted variance $\epsilon_t = \epsilon(x_{t-1})$. Thus, A + C + D is our approach. As before, we omit results for the ablation using the VC generalization bound since n is so small that the bound does not hold for any of the given α and δ. In (b), we show the same ablations over the entire trajectory until $t = 20$. As can be seen, using the calibrated predictor produces a large gain; these gains are most noticeable in the tails. Using the accumulated confidence produces a smaller, but still significant, gain. In (c) and (d), we show how the sizes vary with α and δ, respectively. The trends are similar those for ResNet.

5 Conclusion

We have proposed an algorithm for constructing PAC confidence sets for deep neural networks. Our approach leverages statistical learning theory to obtain theoretical guarantees on the predicted confidence sets. These confidence sets quantify the uncertainty of deep neural networks. For instance, they can be used to inform safety-critical decision-making, and to ensure safety with high-probability in robotics control settings that leverage deep neural networks for perception. Future work includes extending these results to more complex tasks (e.g., structured prediction), and handling covariate shift (e.g., to handle policy updates in reinforcement learning).

Acknowledgments

This work was support in part by NSF CCF-1910769 and by the Air Force Research Laboratory and the Defense Advanced Research Projects Agency under Contract No. FA8750-18-C-0090. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Air Force Research Laboratory (AFRL), the Defense Advanced Research Projects Agency (DARPA), the Department of Defense, or the United States Government.

REFERENCES

Rina Foygel Barber, Emmanuel J Candes, Aaditya Ramdas, and Ryan J Tibshirani. Predictive inference with the jackknife+. arXiv preprint arXiv:1905.02928, 2019.

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In Advances in neural information processing systems, pp. 908–918, 2017.

Marko Bohanec and Vladislav Rajkovic. Knowledge acquisition and explanation for multi-attribute decision making. In 8th Intl Workshop on Expert Systems and their Applications, 1988.

Christopher P Bona de, A Russell Localio, John H Holmes, Vinay M Nadkarni, Shannon Stemler, Matthew MacMurchy, Miriam Zander, Kathryn E Roberts, Richard Lin, and Ron Keren. Video analysis of factors associated with response time to physiologic monitor alarms in a childrens hospital. JAMA pediatrics, 171(6):524–531, 2017.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. arXiv preprint arXiv:1606.01540, 2016.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In Advances in Neural Information Processing Systems, pp. 4754–4765, 2018.

Paulo Cortez and Alice Maria Goncalves Silva. Using data mining to predict secondary school student performance. 2008.

Morris H DeGroot and Stephen E Fienberg. The comparison and evaluation of forecasters. Journal of the Royal Statistical Society: Series D (The Statistician), 32(1-2):12–22, 1983.

Christophe Denis and Mohamed Hebiri. Con dence sets with expected sizes for multiclass classi - cation. The Journal of Machine Learning Research, 18(1):3571–3598, 2017.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 1183–1192. JMLR. org, 2017.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. arXiv preprint arXiv:1706.04599, 2017.

H Altay Guvenir, Burak Acar, Gulsen Demiroz, and Ayhan Cekin. A supervised machine learning algorithm for arrhythmia analysis. In Computers in Cardiology 1997, pp. 433–436. IEEE, 1997.

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 (CVPR), pp. 770–778, 2016.

David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In European Conference on Computer Vision, pp. 749–765. Springer, 2016.

Michael J Kearns and Umesh Virkumar Vazirani. An introduction to computational learning theory. MIT press, 1994.

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. arXiv preprint arXiv:1404.5997, 2014.

Volodymyr Kuleshov and Percy S Liang. Calibrated structured prediction. In Advances in Neural Information Processing Systems, pp. 3474–3482, 2015.

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. arXiv preprint arXiv:1807.00263, 2018.

Jing Lei. Classi cation with con dence. Biometrika, 101(4):755–769, 2014.

Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. Journal of the American Statistical Association, 113 (523):1094–1111, 2018.

Ali Malik, Volodymyr Kuleshov, Jiaming Song, Danny Nemer, Harlan Seymour, and Stefano Ermon. Calibrated model-based deep reinforcement learning. In International Conference on Machine Learning, pp. 4314–4323, 2019.

Allan H Murphy. Scalar and vector partitions of the probability score: Part i. two-state situation. Journal of Applied Meteorology, 11(2):273–282, 1972.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.

Harris Papadopoulos. Inductive conformal prediction: Theory and application to neural networks. In Tools in artificial intelligence. IntechOpen, 2008.

T Pearce, M Zaki, A Brintrup, and A Neely. High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. 35th International Conference on Machine Learning, ICML 2018, volume 9, pp. 6473–6482, 2018.

John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Advances in large margin classifiers, 10(3):61–74, 1999.

J Ross Quinlan. Combining instance-based and model-based learning. In Proceedings of the Tenth International Conference on International Conference on Machine Learning, pp. 236–243. Morgan Kaufmann Publishers Inc., 1993.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3):211–252, 2015.

Lukas Steinberger and Hannes Leeb. Leave-one-out prediction intervals in linear regression models with many variables. arXiv preprint arXiv:1602.05801, 2016.

Lukas Steinberger and Hannes Leeb. Conditional predictive inference for high-dimensional stable algorithms. arXiv preprint arXiv:1809.01412, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In International Conference on Learning Representations (ICLR), 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9, 2015.

Leslie G Valiant. A theory of the learnable. Communications of the ACM, 27(11):1134–1142, 1984.

Vladimir N Vapnik. An overview of statistical learning theory. IEEE transactions on neural networks, 10(5):988–999, 1999.

Vladimir Vovk. Conditional validity of inductive conformal predictors. Machine learning, 92(2-3): 349–376, 2013.

Wenbo Wang and Xingye Qiao. Learning confidence sets using support vector machines. In Advances in Neural Information Processing Systems, pp. 4929–4938, 2018.

Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013.

Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In Proceedings of the Eighteenth International Conference on Machine Learning, Citeseer, 2001.

Bianca Zadrozny and Charles Elkan. Transforming classi er scores into accurate multiclass probability estimates. InProceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 694–699. ACM, 2002.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. ICLR, 2017.

A DISCUSSION OF ALGORITHM DESIGN CHOICES

A.1 USEFULNESS OF TEMPERATURE SCALING

In this section, we discuss why temperature scaling can help improve the predicted confidence sets. A concern is that temperature scaling does not change the ordering of label probabilities. Thus, we may expect that temperature scaling does not affect the predicted confidence sets. However, this fact only holds when considering a single input x—i.e., the ordering of the probabilities $f(y \mid x)$ for $y \in \mathcal{Y}$ is not changed by temperature scaling. Indeed, the order of confidences for labels for different inputs can change. For a concrete example, consider two inputs x and x', and the case $\mathcal{Y} = \{0, 1, 2\}$. Assume that the label probabilities are

$$f(\cdot \mid x) = [1/3 \quad 1/3 \quad 1/3]^\top$$
$$f(\cdot \mid x') = [3/4 \quad 1/4 \quad 0]^\top.$$

Now, if we take temperature very large, then the labels become roughly

$$f_T(\cdot \mid x) = [1/3 \quad 1/3 \quad 1/3]^\top$$
$$f_T(\cdot \mid x') = [1/2 \quad 1/2 \quad 0]^\top.$$

As a consequence, there are confidence sets that are achievable when using f_T that are not achievable when using f. In particular, the confidence sets

$$C_T(x) = ?$$
$$C_T(x') = \{0, 1\}$$

can be achieved using f_T (e.g., with $e^T = 2.5$). However, it is impossible to achieve these confidence sets using f for any choice of T, since if $1 \in C_T(x')$, then it must be the case that $C_T(x) = \{0, 1, 2\}$. Intuitively, we expect calibrated prediction to improve the ordering of probabilities across different inputs. Our experiments support this intuition, since they show that empirically, using calibrated predictors f_T produces confidence sets of smaller size.

A.2 USEFULNESS OF DIRECT BOUND

One key design choice is to use a specialized generalization bound that directly provides PAC guarantees on our confidence sets rather than simply applying the VC dimension bound. The easiest way to determine which bound is better is to examine which one produces a smaller confidence set. In our approach, the size of the confidence set decreases monotonically with the choice of $\epsilon(n, \alpha, \delta)$ in (3). Thus, the bound that produces larger ϵ is better. Recall that the VC dimension bound produces

$$\epsilon_{\text{VC}}(n, \alpha, \delta) = \sqrt{\frac{\log(2n) + 1 - \log(\alpha/4)}{n}};$$

whereas our direct bound produces (for $k = 0$)

$$\epsilon_{\text{direct}}(n, \alpha, \delta) = \max_{k \in \mathbb{N}} k/n \quad \text{subj. to} \quad \sum_{i=0}^{k} \binom{n}{i} \alpha^i (1-\alpha)^{n-i} < \delta.$$

Directly comparing these two choices of ϵ is difficult, but our experiments show empirically that using the direct bound outperforms using the indirect bound.

A more direct way to compare the two approaches is to instead ask how large n needs to be to achieve $\epsilon(n, \alpha, \delta) = 0$. For ϵ_{VC}, it is easy to check that we need

$$n \geq \frac{\log(2n) + 1 + \log(4/\alpha)}{\alpha^2}.$$

Thus, we need n to be at least $O(\log(1/\alpha)/\alpha^2)$ (and possibly greater, to account for the $\log(2n)$ term). In contrast, for our direct bound, $\epsilon = 0$ corresponds to the case $k = 0$. To achieve $k = 0$, it suffices to have n satisfying $(1-\alpha)^n < \delta$. Using $(1-\alpha)^n \leq e^{-n\alpha}$, it suffices to have n satisfying

$$n \geq \frac{\log(1/\delta)}{\alpha}.$$

Figure 4: Sample complexity of different bounds; we fix $\delta = 10^{-5}$. Left: Sample complexity of VC bound and direct bound when $k = 0$. Right: Sample complexity of direct bound for varying k.

In other words, n only needs to be $O(\log(1/\delta)/\varepsilon)$. For small δ (e.g., $\delta = 0.01$), we need 100 fewer samples to achieve the same size confidence set (i.e., with choice $\varepsilon(n, \delta) = 0$). In Figure 4 (right), we compute the exact values of n needed to get $\varepsilon(n, \delta, \hat{f}) = 0$ as a function of δ for each bound (fixing $\delta = 10^{-5}$). As expected, our bound requires substantially smaller n.

Finally, in Figure 4 (right), we compare the magnitude of n needed to achieve larger values of k using our direct bound; for simplicity, we actually consider larger values of γ (where $\gamma = k/n$), but the qualitative insights are the same. As can be seen, even for large k (e.g., $k = 50$), the number of samples increases, but not substantially.

B Theoretical Guarantees

B.1 Assumptions

We make two additional technical assumptions in Theorem 1, both of which are standard. First, we assume that f is measurable; this assumption holds for all models used in practice, including neural networks (e.g., it holds as long as f is continuous).

Second, letting $\ell : Z \to \mathbb{R}$, where $Z = X \times Y$, be defined by $\ell((x, y)) = -\log f(y \mid x)$, we assume that the distribution D induced by ℓ on $\mathbb{R}$ has continuous cumulative distribution function (CDF). More precisely, letting F_D be the measure defining D, then D is defined by the measure

$$F_D(t) = F_D(\ell^{-1}(t)),$$

where $\ell^{-1} : \mathbb{R} \to 2^Z$ is the inverse of ℓ in the sense that $z \in \ell^{-1}(\ell(z))$ for all $z \in Z$. Then, we assume that the CDF corresponding to D is continuous. This second assumption is standard in statistical learning theory (Kearns & Vazirani, 1994). Essentially, it says that for any $t \in \mathbb{R}$, the probability that $t = -\log f(y \mid x)$ must equal zero. This assumption should hold unless $p(x, y)$ or $f(y \mid x)$ are degenerate in some way. Furthermore, we can detect this case. In particular, the failure mode corresponds to the case that we see multiple points with the same value of $f(y \mid x)$. Thus, choosing $\hat{f} = -\log f(y \mid x)$ would include all these points, so the realized error rate is larger than desired for $\hat{f}$. In this case, we can simply choose a slightly larger $\hat{f}$ to avoid this problem.

B.2 Proof of Theorem 1

At a high level, our proof proceeds in three steps. First, we show that a confidence set predictor C_T can be encoded as a binary classifier M_T. Second, we show that a PAC bound for M_T implies a PAC bound for C_T (where in both cases, the unknown parameter $T \in \mathbb{R}$). Third, we prove PAC bounds on the error of $M_{\hat{T}}$; by the second step, these bounds complete our proof.

Encoding C_T as a binary classifier M_T. We begin by showing how the problem of learning a PAC confidence set predictor C_T reduces to the problem of learning a PAC binary classifier M_T. First, we show that for any $T \in \mathbb{R}$, the confidence set predictor C_T can be encoded as a binary classifier M_T. Consider any parameter $T \in \mathbb{R}$. Recall that we use the model $f(y \mid x)$ to construct the confidence set predictor

$$C_T(x) = \{y \in Y \mid f(y \mid x) \geq e^{-T}\}.$$

Now, define the map $\ell : \mathcal{Z} \to \mathbb{R}$ by $\ell(x, y) = -\log f(y \mid x)$, where $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, and define the binary classifier $M_T : \mathbb{R} \to \{0, 1\}$ by

$$M_T(t) = \mathbb{I}[t \leq T].$$

Here, $\mathbb{I}[s]$ is the indicator function, which returns one if a statement s is true and zero otherwise. We claim that

$$C_T(x) = \{y \in \mathcal{Y} \mid M_T(\ell(x, y)) = 1\}. \tag{12}$$

To see this claim, note that

$$\begin{aligned} C_T(x) &= \{y \in \mathcal{Y} \mid f(y \mid x) \geq e^{-T}\} \\ &= \{y \in \mathcal{Y} \mid -\log f(y \mid x) \leq T\} \\ &= \{y \in \mathcal{Y} \mid \ell(x, y) \leq T\} \\ &= \{y \in \mathcal{Y} \mid \mathbb{I}[\ell(x, y) \leq T] = 1\} \\ &= \{y \in \mathcal{Y} \mid M_T(\ell(x, y)) = 1\}, \end{aligned}$$

as claimed.

PAC bound for M_T implies PAC bound for C_T. Next, we show that a PAC bound for M_T implies a PAC bound for C_T. More precisely, we design a data distribution $\tilde{D}$ and loss $\tilde{\ell}$, and show that (i) the distribution of $\tilde{T}$ (trained to optimize M_T) is the same as the distribution of $\hat{T}$ (constructed using our algorithm), and (ii) a PAC bound for $M_{\tilde{T}}$ (where $\tilde{T}$ is trained on data from $\tilde{D}$) implies a PAC bound for $C_{\tilde{T}}$. We show that as a consequence, a PAC bound on $M_{\hat{T}}$ implies a PAC bound on $C_{\hat{T}}$.

We begin by constructing $\tilde{D}$ and $\tilde{\ell}$. To this end, recall that D is a given distribution over $\mathcal{X} \times \mathcal{Y}$. We define a data distribution $\tilde{D}$ over $\tilde{\mathcal{X}} \times \tilde{\mathcal{Y}}$, where $\tilde{\mathcal{X}} = \mathbb{R}$ and $\tilde{\mathcal{Y}} = \{0, 1\}$, as follows. The first component of $\tilde{D}$ is the distribution over $\tilde{\mathcal{X}}$ induced by ℓ from D, and the second component is the distribution over $\tilde{\mathcal{Y}}$ that places all probability mass on 1. Formally, $\tilde{D}$ exists assuming ℓ is measurable, so the induced distribution exists; for all our choices of ℓ (i.e., categorical or Gaussian), this property is satisfied. Then,

$$\tilde{D}((t, a)) = D_{\ell}(\ell^{-1}(t)) \cdot \mathbb{I}[a = 1],$$

where D_{ℓ} is the measure encoding ℓ, and D is the measure encoding D. Furthermore, we define $\tilde{\ell} : \tilde{\mathcal{Y}} \times \tilde{\mathcal{Y}} \to \{0, 1\}$ to be the 0-1 loss $\tilde{\ell}(a, a') = \mathbb{I}[a \neq a']$. Finally, let $\hat{T}$ be chosen using our algorithm—i.e.,

$$\begin{aligned} \hat{T} &= \arg\min T \quad \text{subj. to } L(C_T; Z) \\ L(C_T; Z) &= \frac{1}{|Z|} \sum_{(x,y) \in Z} \mathbb{I}[y \notin C_T(x)], \end{aligned}$$

for any $\alpha \in \mathbb{R}_{\geq 0}$, and let $\tilde{T}$ be chosen similarly for M_T—i.e.,

$$\begin{aligned} \tilde{T} &= \arg\min T \quad \text{subj. to } L(M_T; \tilde{Z}) \\ L(M_T; \tilde{Z}) &= \frac{1}{|\tilde{Z}|} \sum_{(t,a) \in \tilde{Z}} \tilde{\ell}(M_T(t), a) = \frac{1}{|\tilde{Z}|} \sum_{(t,a) \in \tilde{Z}} \mathbb{I}[M_T(t) \neq a]. \end{aligned}$$

Now, we show (i) above. In particular, we claim that $\hat{T}(Z)$ has the same distribution as $\tilde{T}(\tilde{Z})$, where $Z \sim D^n$ and $\tilde{Z} \sim \tilde{D}^n$ are random datasets. To this end, define $\phi : \mathcal{Z}^n \to \tilde{\mathcal{Z}}^n$ by

$$\phi((z_1, \ldots, z_n)) = ((\ell(z_1), 1), \ldots, (\ell(z_n), 1)).$$

Note that

$$\begin{aligned} \tilde{L}(M_T; \phi(Z)) &= \frac{1}{|\phi(Z)|} \sum_{i=1}^{n} \mathbb{I}[M_T(\ell(x_i, y_i)) \neq 1] \\ &= \frac{1}{|Z|} \sum_{i=1}^{n} \mathbb{I}[y_i \notin C_T(x_i)] \\ &= L(C_T; Z), \end{aligned}$$

from which it follows that

$$\begin{aligned}\hat{T}(Z) &= \arg\min_T T \quad \text{subj. to } L(C_T; Z) \\ &= \arg\min_T T \quad \text{subj. to } \bar{L}(M_T; \pi(Z)) \\ &= T(\pi(Z)).\end{aligned}$$

By construction of π, the random variables Z and $\pi(Z)$ have the same distribution; thus, it follows that the random variables $\bar{L}(M_T)$ and $\bar{L}(M_{T(\pi(Z))})$ have the same distribution as well. Since $\hat{T}(Z) = T(\pi(Z))$, it follows that $\hat{T}(Z)$ has the same distribution as $T(Z)$, as claimed.

Next, we show (ii) above. In particular, we claim that a PAC bound for $M_{T(Z)}$—i.e.,

$$P_{Z \sim D^n}[\bar{L}(M_{T(Z)}) \quad] \quad 1 \quad ;$$

implies a PAC bound for $C_{T(Z)}$—i.e.,

$$P_{Z \sim D^n}[L(C_{T(Z)}) \quad] \quad 1 \quad ;$$

where the true losses are

$$\begin{aligned}\bar{L}(M_T) &= E_{(t;a) \sim D}[\ell(M_T(t); a)] = P_{(t;a) \sim D}[M_T(t) \neq a] \\ L(C_T) &= E_{(x;y) \sim D}[\mathbb{I}[y \neq C_T(x)]] = P_{(x;y) \sim D}[y \neq C_T(x)].\end{aligned}$$

Note that it suffices to show that the true loss for C_T equals the true loss for M_T—i.e.,

$$L(C_T) = \bar{L}(M_T);$$

since this equation (together with the PAC bound for $M_{T(Z)}$) implies

$$P_{Z \sim D^n}[L(C_{T(Z)}) \quad] = P_{Z \sim D^n}[\bar{L}(M_{T(Z)}) \quad] \quad 1 \quad ;$$

as desired. To see the claim, note that

$$\begin{aligned}\bar{L}(M_T) &= P_{(t;a) \sim D}[M_T(t) \neq a] \\ &= \int \mathbb{I}[M_T(t) \neq a] \, d\, D((t;a)) \\ &= \sum_{a=0}^{1} \mathbb{I}[a = 1] \int \mathbb{I}[M_T(t) \neq a] \, d\, D(\pi^{-1}(t)) \\ &= \int \mathbb{I}[M_T(t) \neq 1] \, d\, D(\pi^{-1}(t))\end{aligned}$$

Now, using the change of variables $t = \pi(z)$, we have

$$\begin{aligned}\bar{L}(M_T) &= \int \mathbb{I}[M_T(\pi(z)) \neq 1] \, d\, D(z) \\ &= \int \mathbb{I}[M_T(\pi(x;y)) \neq 1] \quad D(x;y)\,dx\,dy.\end{aligned}$$

Then, using (12), we have

$$\begin{aligned}\bar{L}(M_T) &= \int \mathbb{I}[y \neq C_T(x)] D(x;y)\,dx\,dy \\ &= P_{(x;y) \sim D}[y \neq C_T(x)] \\ &= L(C_T);\end{aligned}$$

as claimed.

Finally, combining (i) and (ii), we have

$$P_{Z \sim D^n}[L(C_{\hat{T}(Z)}) \quad] = P_{Z \sim D^n}[L(C_{T(Z)}) \quad] \quad 1 \quad ;$$

where the first equality follows since (i) says that $\hat{A}(Z)$ (where $Z \sim D^n$) has the same distribution as $\tilde{T}(\tilde{Z})$ (where $\tilde{Z} \sim D^n$), and the second inequality follows by (ii).

Generalization bound. Finally, we prove the PAC bound

$$P_{\tilde{Z} \sim D^n}[\tilde{L}(M_{\tilde{T}}) \quad] \quad 1 \quad _0; \tag{13}$$

for $M_{\tilde{T}}$, where $_0 = \sum_{i=0}^{k} \binom{n}{i} (1 \quad)^{n \ i}$; for conciseness, we have dropped the dependence of $\tilde{T}$ on $\tilde{Z}$. By the previous step, this bound implies the theorem statement. To this end, we first simplify the left-hand side of the inequality (13). In particular, let T be the smallest t for which $\tilde{L}(M_T) = \ $; such a T exists by our assumption that P_t has continuous density function.

First, we claim that $T < T$ implies $\tilde{L}(M_T) > \tilde{L}(M_T)$. Assuming $T < T$, then

$$\begin{aligned}
\tilde{L}(M_T) &= P_{(t;a) \ D}[M_T(t) \neq a] \\
&= E_{(t;a) \ D}[I[M_T(t) \neq a] \\
&= E_{(t;a) \ D}[I[M_{\hat{T}}(t) \neq 1]] \\
&= E_{(t;a) \ D}[I[I[t \quad T] \neq 1]] \\
&= E_{(t;a) \ D}[I[t > T \ ]] \\
&> E_{(t;a) \ D}[I[t > T \quad]] \\
&= \tilde{L}(M_T \ ).
\end{aligned}$$

Assuming $T \quad T$, we can similarly show that $\tilde{L}(M_{\hat{T}}) \quad \tilde{L}(M_T)$. It follows that

$$\begin{aligned}
P_{\tilde{Z} \sim D^n}\left[\tilde{L}(M_{\tilde{T}}) > \quad \right] &= P_{\tilde{Z} \sim D^n}\left[\tilde{L}(M_{\tilde{T}}) > \tilde{L}(M_T \ )\right] \\
&= P_{\tilde{Z} \sim D^n}\left[\tilde{T} < T \quad \right].
\end{aligned}$$

As a consequence, (13) is equivalent to

$$P_{\tilde{Z} \sim D^n}\left[\tilde{T} < T \quad \right] \quad _0.$$

Next, recall that $\tilde{T}$ must satisfy $\tilde{L}(M_{\tilde{T}}; \tilde{Z}) \quad$, where

$$\tilde{L}(M_{\tilde{T}}; \tilde{Z}) = \frac{1}{n} \sum_{(t;a) \in \tilde{Z}} I[M_{\tilde{T}}(t) \neq a].$$

Assuming $\hat{T} < T$, and using $k = n \quad$, it follows that

$$\begin{aligned}
k \quad &\sum_{(t;a) \in \tilde{Z}} I[M_{\tilde{T}}(t) \neq a] = \sum_{(t;a) \in \tilde{Z}} I[M_{\tilde{T}}(t) \neq 1] \\
&= \sum_{(t;a) \in \tilde{Z}} I[t > \tilde{T}] \\
&\quad \sum_{(t;a) \in \tilde{Z}} I[t > T \ ].
\end{aligned}$$

As a consequence, we have

$$\begin{aligned}
P_{\tilde{Z} \sim D^n}\left[\tilde{T} < T \quad \right] &\quad P_{\tilde{Z} \sim D^n}\left[4 \sum_{(t;a) \in \tilde{Z}} I[t > T \ ] \quad k5\right] \\
&= \sum_{i=0}^{k} P_{\tilde{Z} \sim D^n}\left[4 \sum_{(t;a) \in \tilde{Z}_{val}} I[t > T \ ] = i5\right].
\end{aligned}$$

By our definition of T^*, the event in the final expression says that the sum of n i.i.d. Bernoulli random variables $\mathbb{1}[t > T^*]$ Bernoulli(α) is at most k. Thus, this event follows a distribution Binomial$(n; \alpha)$, so

$$P_{Z^* \sim D^n}\left[T^* < T\right] = \sum_{i=0}^{k} \text{Binomial}(i; n; \alpha) = \sum_{i=0}^{k} \binom{n}{i} \alpha^i (1-\alpha)^{n-i} = \beta;$$

as claimed. The theorem statement follows.

C Details on Probability Forecasters for Specific Tasks

In this section, we describe architectures for probability forecasters for classification, regression, and model-based reinforcement learning.

Classification. For the case $Y = \{1, \ldots, Y\}$, we choose the probability forecaster f to be a neural network with a softmax output. Then, we can compute a given confidence set

$$C_T(x) = \{y \in Y \mid f(y \mid x) \geq e^{-T}\}$$

by explicitly enumerating $y \in Y$. We measure the size of $C_T(x)$ as $|C_T(x)|$.

Regression. For the case $Y = \mathbb{R}$, we choose the probability forecaster f to be a neural network that outputs the parameters $(s, \sigma) \in Y \times \mathbb{R}_{>0}$ of a Gaussian distribution. Then, we have

$$C_T(x) = \left[s - \sigma\sqrt{2(T - \log(\sigma\sqrt{2}))},\ s + \sigma\sqrt{2(T - \log(\sigma\sqrt{2}))}\right].$$

This choice generalizes to $Y = \mathbb{R}^d$ by having f output the parameters $(s, \Sigma) \in Y \times S^d_{\succ 0}$ (where $S^d_{\succ 0}$ is the set of d dimensional symmetric positive definite matrices) of a d dimensional Gaussian distribution. Note that $C_T(x)$ is an ellipsoid $C_T(x) = \mu + \Sigma S^{d-1}$, where $\mu \in \mathbb{R}^d$ and S^{d-1} is the unit sphere in $\mathbb{R}^d$; in particular, $\Sigma = D^{\frac{1}{2}} Q$, where $Q D Q^\top$ is the eigendecomposition of

$$(2T - d \ln 2 - \ln \det \Sigma)^{-1} \Sigma^{-1}.$$

We measure the size of $C_T(x)$ as $\|\Sigma\|_F$, where $\|\cdot\|_F$ is the Frobenius norm.

Model-based reinforcement learning. In model-based reinforcement learning, the goal is to predict trajectories based on a model of the dynamics. We consider an MDP with X states $\subseteq \mathbb{R}^{d_X}$, actions $U \subseteq \mathbb{R}^{d_U}$, an unknown distribution over initial states $x_0 \sim d_0$, and unknown dynamics $g(x' \mid x, u)$ mapping a state-action pair $(x, u) \in X \times U$ to a distribution over states $x' \in X$. We assume a fixed, known policy $\pi(u \mid x)$, mapping a state $x \in X$ to a distribution over actions $u \in U$. The (unknown) closed-loop dynamics are $f(x' \mid x) = \mathbb{E}_{u \sim \pi(u \mid x)}[g(x' \mid x, u)]$.

Given initial state $x_0 \in X$ and time horizon $H \in \mathbb{N}$, we can sample a trajectory $x_{1:H} = (x_1, \ldots, x_H) \sim f$ by setting $x_0 = x_0$ and sequentially sampling $x_{t+1} \sim f(\cdot \mid x_t)$ for $t \in \{0, 1, \ldots, H-1\}$. Our goal is to predict a confidence set $C_T(x_0) \subseteq X^H$ that contains $x_{1:H} \in X^H$ with high probability (according to both the randomness in initial states $x_0 \sim d_0$ and in f). This problem is a multivariate regression problem with inputs X and outputs $Y = X^H$.

We assume given a probability forecaster $\pi(x' \mid x) = \mathcal{N}(x'; \mu(x), \Sigma(x))$ trained to predict the distribution over next states—i.e., $\pi(x' \mid x) \approx f(x' \mid x)$. Given initial state $x_0 \in X$ and time horizon $H \in \mathbb{N}$, we construct the mean trajectory $x_{1:H}$ by setting $x_0 = x_0$ and letting $x_{t+1} = \mu(x_t)$. To account for the fact that the variances accumulate over time, we sum them together to obtain the predicted variances $\tilde{\Sigma}_{1:H}$—i.e.,

$$\tilde{\Sigma}_t = \Sigma(x_0) + \Sigma(x_1) + \cdots + \Sigma(x_{t-1}).$$

Then, we use the probability forecaster $\mathcal{N}(x_{1:H}; \tilde{\Sigma}_{1:H}) = \mathcal{N}(x_{1:H}; \tilde{\Sigma}_{1:H})$ (where we think of $x_{1:H}$ as a vector in $\mathbb{R}^{H d_X}$ and $\tilde{\Sigma}_{1:H}$ as a block diagonal matrix in $\mathbb{R}^{(H d_X) \times (H d_X)}$) to construct confidence sets.

Finally, we describe how we measure the size of a predicted confidence set $C_T(x_0) \subseteq X^H$. In particular, note that $C_T(x_0)$ has the form

$$C_T(x_0) = (C_{T;1}(x_0); \ldots; C_{T;H}(x_0));$$

(a) (b)

(c) (d)

Figure 5: Comparison to baselines that do not have theoretical guarantees. In (a) and (b), we show results for ImageNet, and in (c) and (d), we show results for the half-cheetah. In (a) and (c), we show the empirical error in the con dence set sizes; the dotted line denotes $\alpha = 0.01$, our target con dence set error. In (b) and (d), we show the sizes of the constructed con dence sets.

i.e., $C_{T,t}(x_0)$ is the con dence set for the state reached after t time steps. Then, we measure the size of the con dence set for each component, $C_{T,t}(x_0)$ (for $t \in \{1, \ldots, H\}$) individually, and take the average. As in the case of regression, $C_{T,t}(x_0)$ is an ellipsoid $C_{T,t}(x_0) = x_t + {}_t S^{d_x - 1}$; then, the size of $C_T(x_0)$ is $H^{-1} \sum_{t=1}^{H} \| {}_t \|_F$.

An additional detail is that when we calibrate this forecaster, we calibrate each component $C_{T,t}(x_0)$ individually—i.e., we use H calibration parameters $\hat{}_1, \ldots, \hat{}_H$.

D Additional Results

D.1 Comparison to Additional Baselines

We compare to two baselines that do not have theoretical guarantees. We assume given a probability forecaster $f(y \mid x)$. Then, given an input $x \in X$, we construct the con dence set to satisfy

$$\sum_{y \in C(x)} f(y \mid x) \geq 1 - . \tag{14}$$

More precisely, we rst rank the labels in decreasing order of $f(y \mid x)$, to obtain a list $(y_1, y_2, \ldots, y_{|Y|})$. Then, we choose the smallest k such that (14) holds for $C(x) = \{y_1, \ldots, y_k\}$. Intuitively, if the probabilities $f(y \mid x)$ are correct (i.e., $f(y \mid x)$ is the true probability of y given x), then this con dence set should contain the true label y with high probability.

For regression, we cannot explicitly rank labels $y \in Y = \mathbb{R}^d$, but they are monotonically decreasing away from the mean. Then, assuming $f(y \mid x) = N(y; (x); (x))$ is Gaussian, we take an ellipsoid of shape (x) around (x) with minimum radius that captures $1 - $ of the probability mass of $f(y \mid x)$. More precisely, we choose

$$C(x) = C_{\hat{T}(x)}(x)$$
$$\hat{T}(x) = \arg\min_{T \in \mathbb{R}} T \quad \text{subj. to } P_{f(y|x)}[y \in C_T(x)] \geq 1 - ;$$

Figure 6: Con dence set sizes for two neural network architectures trained on ImageNet; for both, we use $n = 20{,}000$, $\alpha = 0.01$ and $\delta = 10^{-5}$. Left: AlexNet (Krizhevsky, 2014); here, the empirical con dence set error of our approach $C_{\hat{T}} + D$ is 0.0066. Right: GoogLeNet (Szegedy et al., 2015); here, the empirical con dence set error of our approach is 0.0061.

where $C_T(x) = \{y \in Y \mid \hat{f}(y \mid x) \geq e^{-T}\}$ as before. Note that unlike our algorithm, the threshold $\hat{T}(x)$ is not a learned parameter, but is computed independently for each new input x. We can solve for $\hat{T}(x)$ ef ciently by changing basis to convert $\hat{f}(y \mid x)$ to a standard Gaussian distribution, and then using the error function to compute the cutoff that includes the desired probability mass.

In Figure 5, we compare the con dence sets constructed using this approach with (i) the forecaster $\hat{f}(y \mid x)$ without any calibration, and (ii) the calibrated forecaster, $\hat{f}_{\text{cal}}(y \mid x)$. We plot both the con dence set sizes and the empirical error rates. For the latter, recall that a con dence set predictor C is correct if $L(C) < \alpha$, where $L(C)$ the true error rate. However, we cannot measure $L(C)$; instead, we approximate it on a held-out test set $Z_{\text{test}} \subseteq X \times Y$—i.e., $L(C) \approx \hat{L}(C; Z_{\text{test}})$, where

$$\hat{L}(C; Z_{\text{test}}) = \frac{1}{|Z_{\text{test}}|} \sum_{(x,y) \in Z_{\text{test}}} \mathbb{I}[y \notin C(x)].$$

Intuitively, $\hat{L}(C; Z_{\text{test}})$ is the fraction of inputs $(x, y) \in Z_{\text{test}}$ such that the predicted con dence set for x does not contain y. We say a con dence set C is empirically valid when $\hat{L}(C; Z_{\text{test}}) < \alpha$. Recall that our algorithm guarantees correctness with probability at least $1 - \delta$, where $\delta = 10^{-5}$.

As can be seen, the baseline approaches are not empirically valid in all cases. In one case—namely, the baseline with the calibrated forecaster on ImageNet—the con dence sets are almost empirically valid. However, in this case, the con dence sets are much larger than those based on our approach, despite the fact that the error rate of our con dence sets are empirically valid. Thus, our algorithms outperform the baselines in all cases.

D.2 Results on Additional ImageNet Neural Network Architectures

We apply our approach to two additional neural network architectures for ImageNet: AlexNet (Krizhevsky, 2014) and GoogLeNet (Szegedy et al., 2015). Our results are shown in Figure 6. As can be seen, calibration reduces the con dence set sizes for AlexNet, but actually increases the con dence set sizes for GoogleNet. Thus, both calibrated and uncalibrated models may need to be considered when constructing con dence set predictors. Also, we nd that con dence set sizes are correlated with classi cation error—the test errors for AlexNet, GoogleNet, and ResNet are 43.38%, 29.41%, and 21.34%, respectively, and their con dence set sizes decrease in the same order.

D.3 Results on Additional Classification Datasets

We apply our approach to three small classi cation datasets: an Arrhythmia detection dataset (Guvenir et al., 1997), a car evaluation dataset (Bohanec & Rajkovic, 1988), and a medical alarm dataset (Bona de et al., 2017). The con dence set sizes are shown in Figure 7. We choose larger values of α and δ since we cannot obtain con dence sets that satisfy the PAC criterion with smaller values when the number of validation examples is too small. For all three datasets, the empirical con dence set error is smaller than the speci ed error; thus, the constructed con dence sets are empirically valid. For these datasets, the con dence set sizes of our approach $C_{\hat{T}} + D$ and our approach without calibration D are similar, most likely due to the small number of class labels.

(a) Arrhythmia (b) Car

(c) CHOP Alarm (d) Suppressed alarms

Figure 7: Con dence set sizes for three additional classi cation benchmarks: (a) the arrhythmia detection dataset (Guvenir et al., 1997); here, $n = 90$, $\alpha = 0.1$, $\delta = 0.05$, and the empirical con dence set error of our approach (C + D) is 0.0435; (b) the car evaluation dataset (Bohanec & Rajkovic, 1988); here, $n = 345$, $\alpha = 0.05$, $\delta = 10^{-5}$, and the empirical con dence set error of our approach (C + D) is 0.0172; and (c) the CHOP alarm dataset (Bona de et al., 2017); here, $n = 1000$, $\alpha = 0.02$, $\delta = 10^{-5}$, and the empirical con dence set error of our approach (C + D) is 0.0159. (d) The fractions of actionable and false alarms with a con dence set of $\{0\}$ (i.e., only contains false alarm).

We additionally ran our approach on a medical dataset where classi cation decisions are safety critical; thus, correct predicted con dence sets are required. In particular, we use the Children's Hospital of Philadelphia (CHOP) alarm dataset (Bona de et al., 2017). This dataset consists of vital signs from 100 patients around one year of age. One of the vital signs is the oxygen level of the blood, and a medical device generates an alarm if the oxygen level is below a speci ed level. The labels indicate whether the generated alarm is true ($y = 1$) or false ($y = 0$). We use $n = 1000$, $\alpha = 0.02$, and $\delta = 10^{-5}$. The empirical con dence set error of our approach is $\hat{L}(C; Z_{\text{test}}) = 0.0159$.

The key question is how many false alarms can be reliably detected using machine learning to help reduce alarm fatigue. We consider an approach where we use the predicted con dence sets to detect false alarms. In particular, we rst train a probability forecaster $X \to P_Y$, where $Y = \{0, 1\}$, to predict the probability that an alarm is true, and then construct a calibrated con dence set predictor $\tilde{f} : X \to 2^Y$ based on this forecaster. We consider an alarm to be false if the predicted con dence set is $\tilde{f}(x) = \{0\}$—i.e., according to our con dence set predictor, the alarm is de nitely false. Then, our PAC guarantee says that the alarm is actually false with probability at least $1 - \alpha$. In summary, we suppress an alarm if $\tilde{f}(x) = \{0\}$. Using our approach, 176=630 (i.e., 27:94%) of false alarms are suppressed, while only 13=187 (i.e., 6:95%) true alarms are suppressed (see Figure 7 (d)).

D.4 Results on Additional Regression Datasets

We ran our algorithm on two small regression baselines—the Auto MPG dataset (Quinlan, 1993) and the student grade dataset (Cortez & Silva, 2008). We show results in Figure 8. The parameters we use are $\alpha = 0.1$ and $\delta = 0.05$; as with the smaller classi cation datasets, we use larger choices of α and δ since we cannot construct valid con dence sets for smaller choices. For the Auto MPG dataset, the empirical con dence set error of our nal model (C + D) is $\hat{L}(C; Z_{\text{test}}) = 0.0597$, so these are empirically valid. For the student grade dataset, the error is $\hat{L}(C; Z_{\text{test}}) = 0.1250$, which is slightly larger than desired; this failure is likely due to the fact that the failure probability $\delta = 0.05$ is somewhat large.

Figure 8: Con dence set sizes for two benchmarks focused on regression; for both, we use $\alpha = 0.1$ and $\delta = 0.05$. Left: the Auto MPG dataset (Quinlan, 1993); here, $n = 70$, and the empirical con dence set error of our approach (C+ D) is 0.1250. Right: The student grade dataset (Cortez & Silva, 2008); here, $n = 100$, and the empirical con dence set error of our approach is 0.0597.

D.5 Additional Results on ImageNet, Half-Cheetah, and Object Tracking

Table 4 & 5 show examples of ResNet con dence set sizes for ImageNet images. Table 6 shows results for varying ; on ResNet. Tables 7 & 8 show results for varying , on the Half-Cheetah. Table 9 shows visualizations of the con dence sets predicted for our object tracking benchmark.

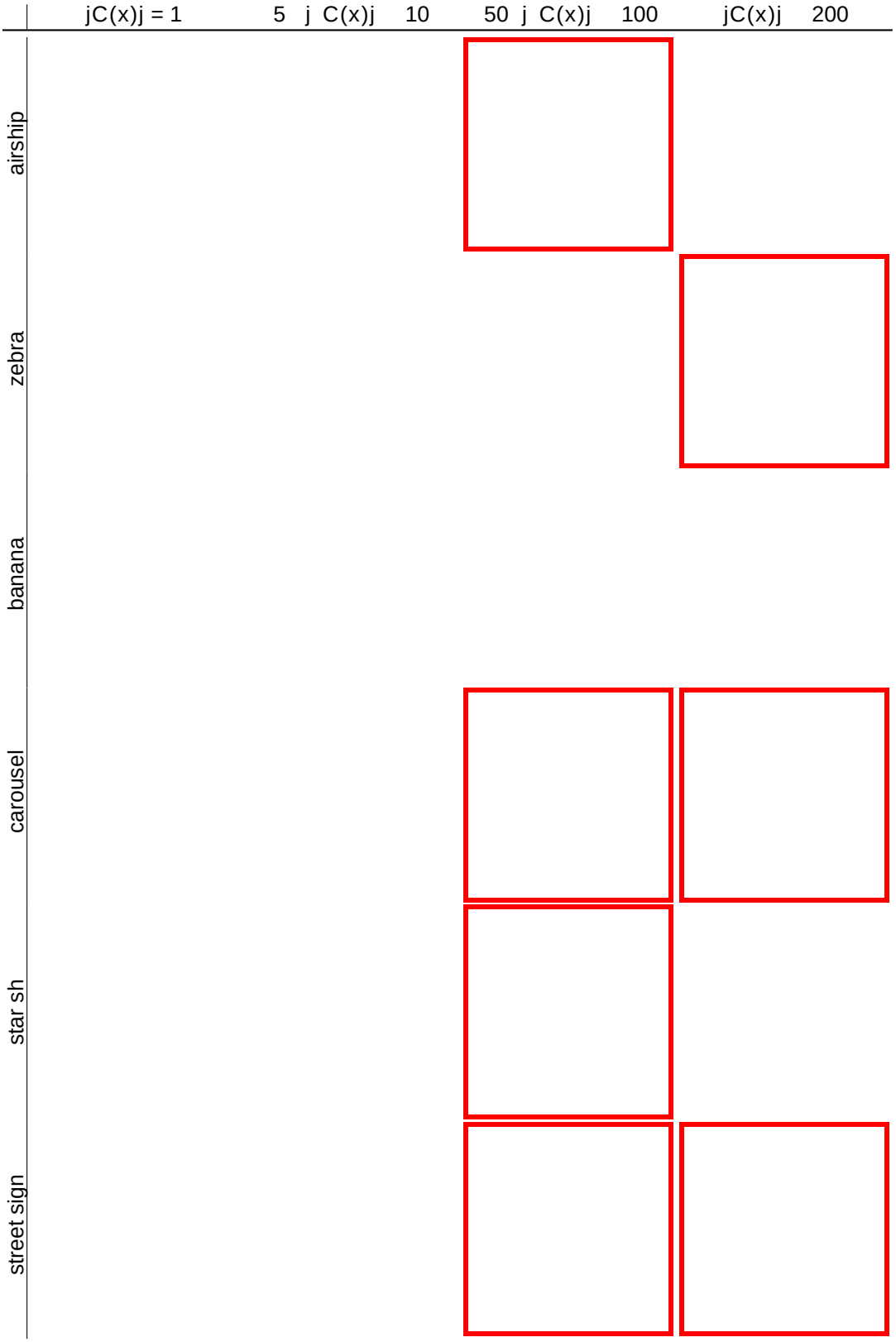


Table 4: ImageNet images with varying ResNet con dence set sizes. The con dence set sizes are on the top. The true label is on the left-hand side. Incorrectly labeled images are boxed in red.

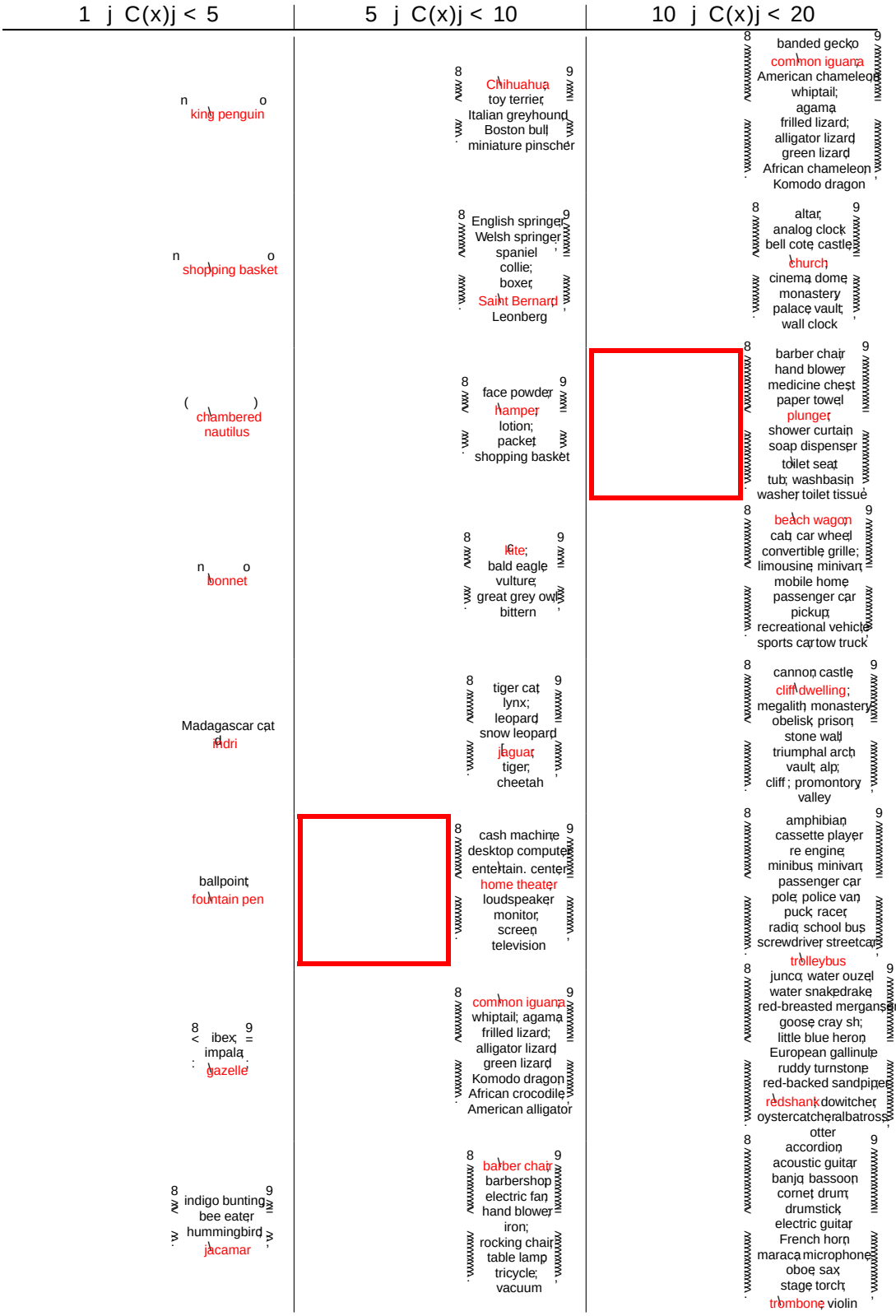


Table 5: Con dence sets of ImageNet images with varying ResNet con dence set sizes. The predicted con dence set is shown to the right of the corresponding input image. The true label is shown in red, and the predicted label is shown with a hat.

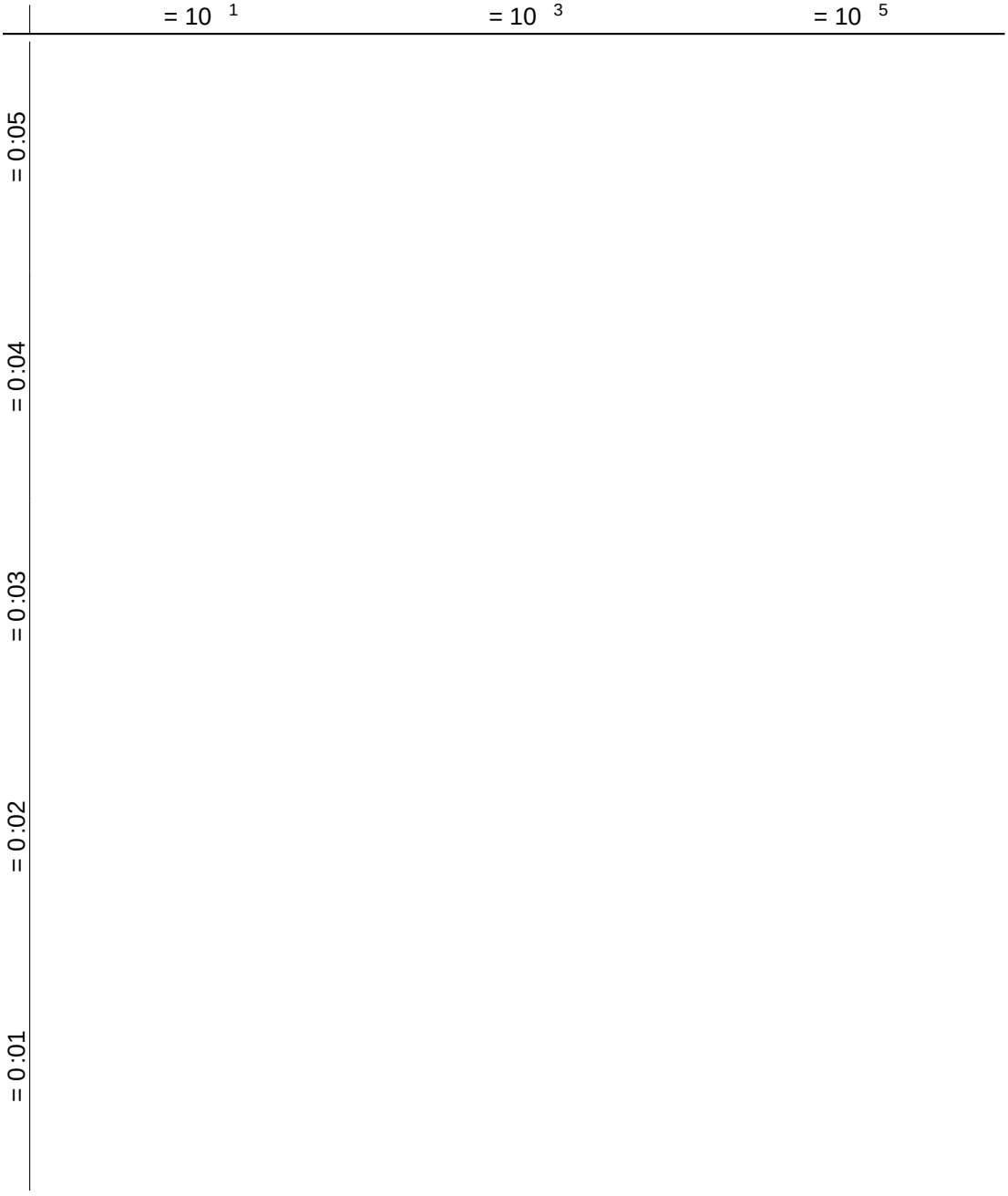


Table 6: Con dence set sizes for ResNet trained on ImageNet, for varying and for $n = 20,000$. The plots are as in Figure 1 (a).

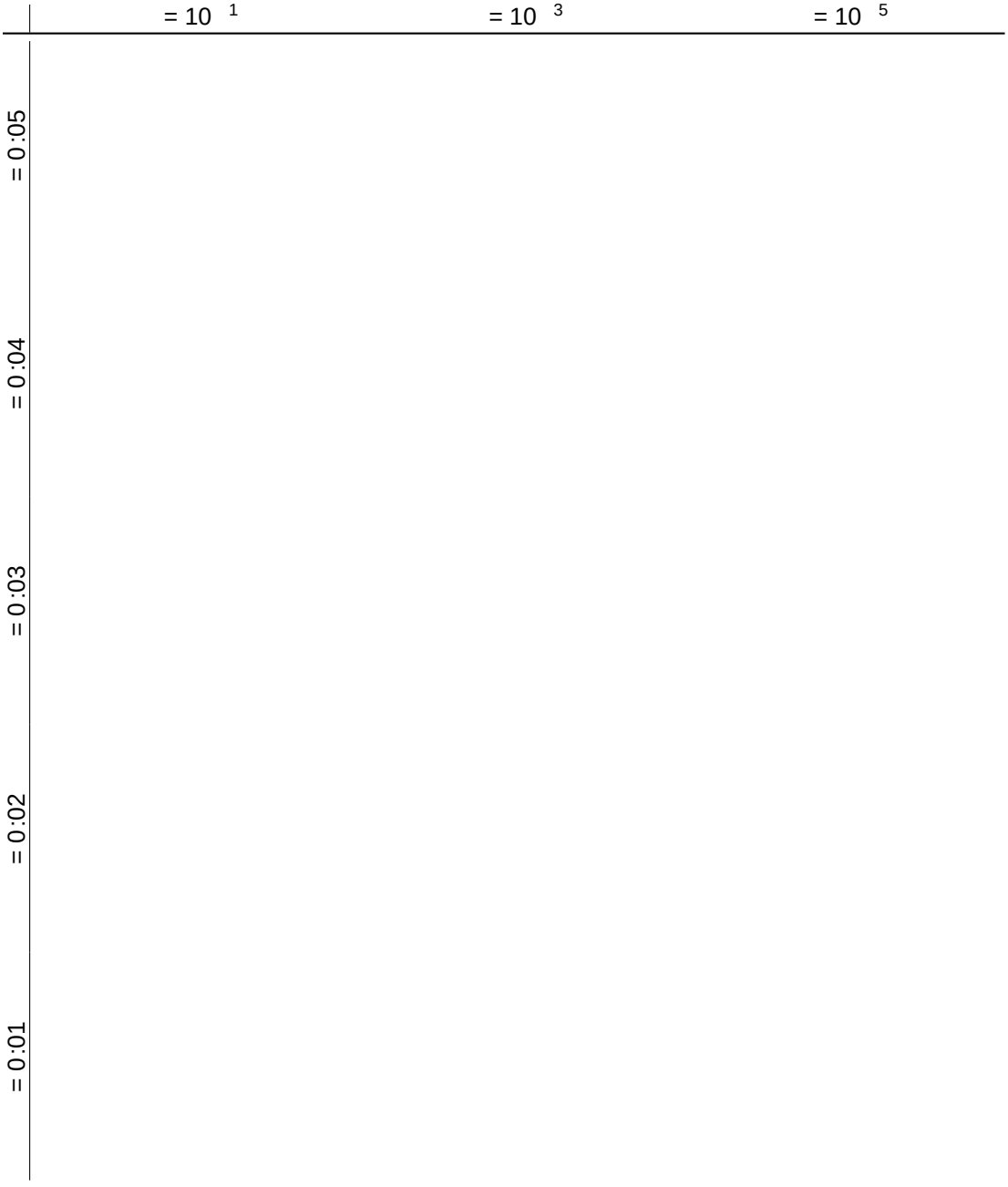


Table 7: Con dence set sizes for a neural network dynamics model trained on the half-cheetah environment, for varying; and forn = 5000. The plots are as in Figure 3 (a).

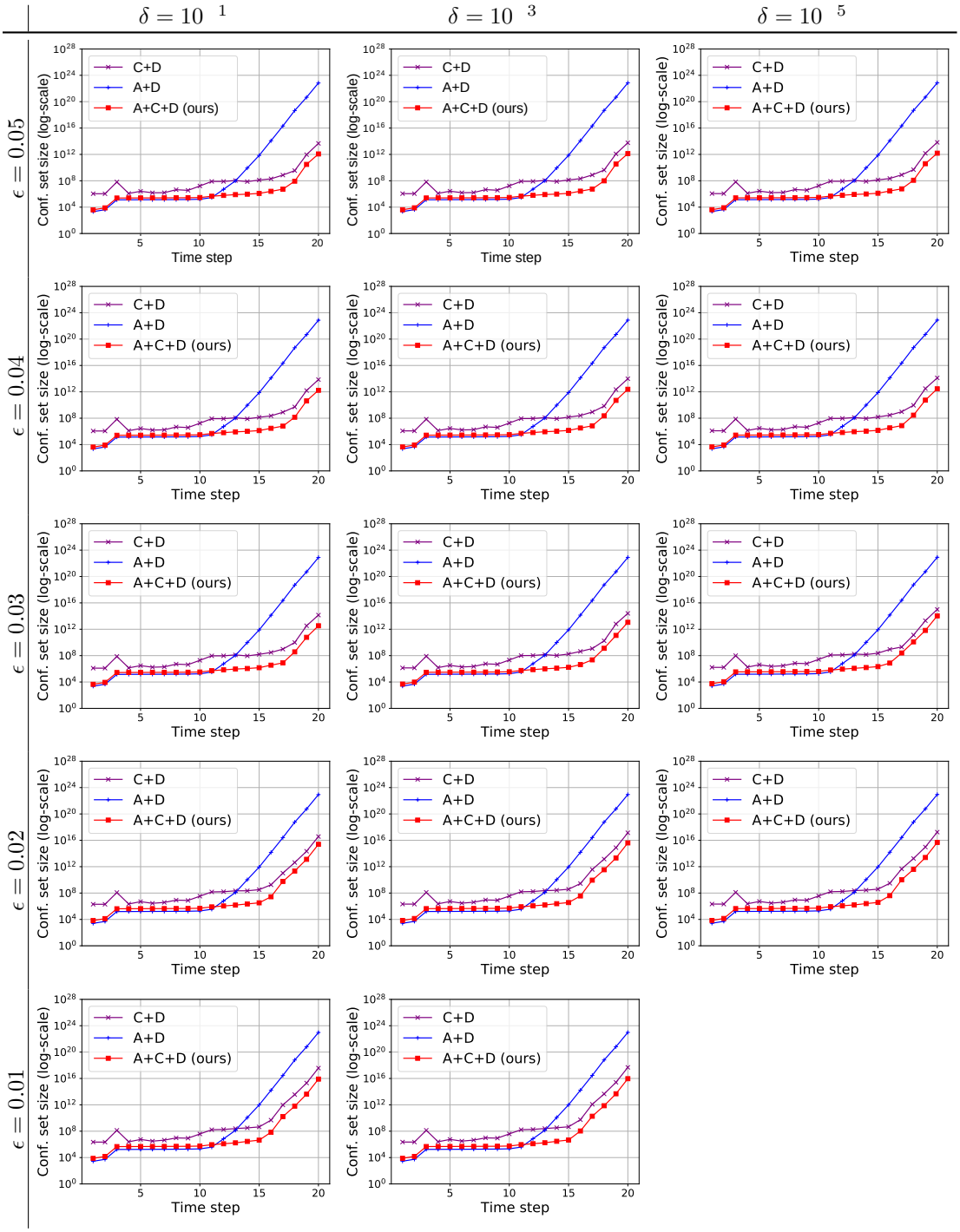


Table 8: Confidence set sizes for a neural network dynamics model trained on the half-cheetah environment, for varying ϵ, δ and for $n = 5000$. The plots are as in Figure 3 (b).