PAC CONFIDENCE SETS FOR DEEP NEURAL NETWORKS VIA CALIBRATED PREDICTION

Anonymous authors
Paper under double-blind review

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, and on a dynamics model the half-cheetah reinforcement learning problem.

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 et al., 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 Z_{val} 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 Z_{val}. 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 T 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 $C_T(x) \subseteq \mathcal{Y}$ (parameterized by $T \in \mathbb{R}_{>0}$) should contain the true label y for at least a $1 - \epsilon$ fraction of cases:

$$\mathbb{P}_{(x,y)\sim D}[y \in C_T(x)] \geq 1 - \epsilon.$$

Since we are fitting 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 $\epsilon, \delta \in \mathbb{R}_{>0}$, we aim to obtain *probably approximately correct (PAC)* confidence sets $C_T(x) \subseteq \mathcal{Y}$ satisfying the guarantee

$$\mathbb{P}_{Z_{\text{val}}\sim D^n}\left(\mathbb{P}_{(x,y)\sim D}(y \in C_T(x)) \geq 1 - \epsilon\right) \geq 1 - \delta.$$

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

	$\|C(x)\| = 1$	$5 \leq \|C(x)\| \leq 10$	$50 \leq \|C(x)\| \leq 100$	$\|C(x)\| \geq 200$
airship				
zebra				

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

Contributions. We propose an algorithm combining calibrated prediction and statistical learning theory to construct PAC confidence sets for deep neural networks (Section 3). We propose instantiations of this framework in the settings of classification, regression, and learning models for reinforcement learning (Section 4). Finally, we evaluate our approach on two benchmarks: ResNet (He et al., 2016) for ImageNet (Russakovsky et al., 2015), and on a probabilistic dynamics model (Chua et al., 2018) learned for the half-cheetah environment (Brockman et al., 2016) (Section 5). Examples of ImageNet images with different sized ResNet confidence sets are shown in Figure 1. As can be seen, our confidence sets become larger and the images become more challenging to classify.

Related work. There has been work on constructing confidence sets with theoretical guarantees. Oftentimes, these guarantees are asymptotic rather than finite sample (Steinberger & Leeb, 2016; 2018). Alternatively, there has been work focused on predicting confidence 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 specific prediction tasks such as binary classification (Lei, 2014; Wang & Qiao, 2018). There has also been work in the setting of regression (Lei et al., 2018; Barber et al., 2019). However, in this case, the confidence sets are fixed 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 confidence sets from a given confidence predictor, and provided PAC guarantees for the validity of these confidence 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 confidence 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 et al., 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 $\mathcal{X}$ be the inputs and $\mathcal{Y}$ be the labels, and let D be a distribution over $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. A *confidence set predictor* is a function $C : \mathcal{X} \to 2^{\mathcal{Y}}$ such that $C(x) \subseteq \mathcal{Y}$ 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

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

Finally, consider an algorithm $\mathcal{A}$ that takes as input a validation set $Z_{\text{val}} \subseteq \mathcal{Z}$ 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 $\mathcal{A}$ is *probably approximately correct (PAC)* if

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

Our goal is to design an algorithm $\mathcal{A}$ 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 $C(x) \subseteq \mathcal{Y} = \{1, ..., Y\}$, 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 $C(x) = [a, b] \subseteq \mathcal{Y} = \mathbb{R}$, 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 $Z_{\text{val}} \subseteq \mathcal{Z}$ of n i.i.d. samples $(x, y) \sim D$, and output the confidence set predictor $C_{\hat{\theta}}$, where $\hat{\theta}$ minimizes the empirical risk:

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

Here, $\mathbb{I}[\phi] \in \{0, 1\}$ is the indicator function, and the empirical risk $\hat{L}$ in 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 $n = |Z_{\text{val}}|$. 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:

$$\hat{\theta} = \operatorname*{arg\,min}_{\theta\in\Theta} S(\theta) \quad \text{subj. to} \quad \hat{L}(C_\theta; Z_{\text{val}}) \leq \alpha. \tag{3}$$

At a high level, the value $\alpha = \alpha(n, \epsilon, \delta) \in \mathbb{R}_{>0}$ 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 et al., 1999), the parameter space Θ 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 Θ, (ii) size metric $S(\theta)$, and (iii) confidence level $\alpha(n, \epsilon, \delta)$ in more detail, as well as how to solve (3) given these choices.

3.1 Choice of Parameter Space Θ

Probability forecasters. Our construction of the parameteric family of confidence set predictors C_θ assumes given a *probability forecaster* $f : \mathcal{X} \to \mathcal{P}_{\mathcal{Y}}$, where $\mathcal{P}_{\mathcal{Y}}$ is a space of probability distributions over $\mathcal{Y}$. Given such an f, we use $f(y \mid x)$ to denote the probability of label y under

Algorithm 1 Algorithm for solving (3).

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

distribution $f(x)$. Intuitively, $f(y \mid x)$ should be the probability (or probability density) that y is the true label for a given input x—i.e., $f(y \mid x) \approx \mathbb{P}_{(X,Y)\sim D}[Y = y \mid X = x]$. For example, in classification, we can choose $\mathcal{P}_{\mathcal{Y}}$ to be the space of categorical distributions over $\mathcal{Y}$, and f may be a neural network whose last layer is a softmax layer with $|\mathcal{Y}|$ outputs. Then, $f(y \mid x) = f(x)_y$. Alternatively, in regression, we can choose $\mathcal{P}_{\mathcal{Y}}$ to be the space of Gaussian distributions, and f may be a neural network whose last layer outputs the values $(\mu, \sigma) \in \mathbb{R} \times \mathbb{R}_{>0}$ of a Gaussian distribution. Then, $f(y \mid x) = \mathcal{N}(x; \mu(x), \sigma(x)^2)$, where $(\mu(x), \sigma(x)) = f(x)$, and $\mathcal{N}(\cdot; \mu, \sigma^2)$ is the Gaussian density function with mean μ and variance σ^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*. First, we consider a parametric model family $\mathcal{F} = \{f_\phi \mid \phi \in \Phi\}$, where Φ is the parameter space. Note that Φ can be high-dimensional—e.g., the weights of a neural network model. Given a training set $Z_{\text{train}} \subseteq \mathcal{Z}$ of m i.i.d. samples $(x, y) \sim D$, the maximum likelihood estimate (MLE) of ϕ is

$$\hat{\phi} = \operatorname*{arg\,min}_{\phi \in \Phi} \ell(\phi; Z_{\text{train}}) \qquad \text{where} \qquad \ell(\phi; Z_{\text{train}}) = -\sum_{(x,y)\in Z_{\text{train}}} \log f_\phi(y \mid x). \tag{4}$$

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

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

More precisely, consider the model family $\mathcal{F}' = \{f_{\hat{\phi},\tau} \mid \tau \in \mathbb{R}_{>0}\}$, where

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

Then, we have the following MLE for τ:

$$\hat{\tau} = \operatorname*{arg\,min}_{\tau \in \mathbb{R}_{>0}} \ell'(\tau; Z'_{\text{train}}) \qquad \text{where} \qquad \ell'(\tau; Z'_{\text{train}}) = -\sum_{(x,y)\in Z'_{\text{train}}} \log f_{\hat{\phi},\tau}(y \mid x). \tag{5}$$

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

Parametric family of confidence set predictors. Finally, given a probability forecaster f, We consider one dimensional parameter space $\Theta = \mathbb{R}_{>0}$; in an analogy to the temperature scaling technique for calibrated prediction, we denote this parameter by $T \in \Theta$. In particular, we assume given a confidence probability predictor f, and consider

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

In other words, $C_T(x)$ is the set of y with high probability given x according to f.

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 $\alpha(n, \epsilon, \delta)$

Naive approach based on VC generalization bound. A naive approach to choosing $\alpha(n, \epsilon, \delta)$ 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 Θ for this problem is 1. Thus, the VC dimension bound implies that for all $T \in \Theta$,

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

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

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

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

$$\mathbb{P}_{Z_{\text{val}} \sim D^n}\left[L(C_{\hat{T}}) > \epsilon\right] < \delta,$$

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

Direct generalization bound. It turns out that since we can actually get better choices of α by directly bounding the generalization error. For instance, in the realizable setting (i.e., we always have $\hat{L}(C_{\hat{T}}; Z_{\text{val}}) = 0$), we can get rates of $O(1/\epsilon)$ instead of $O(1/\epsilon^2)$ (Kearns & Vazirani, 1994). We can achieve these rates by choosing $\alpha = 0$, but then, the PAC guarantees we obtain may actually be stronger than desired (i.e., for $\epsilon' < \epsilon$). Intuitively, we can directly prove a bound that interpolates between the realizable setting and the VC generalization bound. In particular, we have:

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

$$\mathbb{P}_{Z_{val} \sim D^n}\left[L(C_{\hat{T}}) > \epsilon\right] \leq \sum_{i=0}^{k} \binom{n}{i} \epsilon^i (1 - \epsilon)^{n-i},$$

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

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

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

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 $\alpha = \alpha(n, \epsilon, \delta)$ chosen according to (8). Then, we have*

$$\mathbb{P}_{Z_{val} \sim D^n}\left[L(C_{\hat{T}}) > \epsilon\right] < \delta.$$

In other words, our algorithm is probably approximately correct.

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 Θ, $S(T)$, and $\alpha(n, \epsilon, \delta)$ described in the preceding sections. There are two key implementation details that we describe here.

Computing $\alpha(n, \epsilon, \delta)$. To compute $\alpha(n, \epsilon, \delta)$, we need to solve (8). A straightforward approach is to enumerate all possible choices of $k \in \{0, 1, ..., n\}$. 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 $\sum_{i=0}^{k} \binom{n}{i} \epsilon^i (1-\epsilon)^{n-i}$, we can accumulate the sum as we iterate over k. We can also incrementally compute $\binom{n}{i}$, ϵ^i, and $(1-\epsilon)^{n-i}$. For numerical stability, we perform these computations in log space.

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

$$\sum_{(x,y)\in Z_{\text{val}}} E(x, y; T) \leq n \cdot \alpha(n, \epsilon, \delta) \qquad \text{where} \quad E(x, y; T) = \mathbb{I}\left[f_{\hat{\phi},\hat{\tau}}(y \mid x) \geq e^{-T}\right]. \tag{9}$$

Also, note that $k^* = n \cdot \alpha(n, \epsilon, \delta)$ is an integer due to the definition of $\alpha(n, \epsilon, \delta)$ 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 $f_{\hat{\phi},\hat{\tau}}(y \mid x)$ becomes larger. Thus, we can sort the points $(x, y) \in Z_{\text{val}}$ in ascending order of $f_{\hat{\phi},\hat{\tau}}(y \mid x)$, 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

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

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.,

$$\hat{T} = -\log f_{\hat{\phi},\hat{\tau}}(y_{k^*+1} \mid x_{k^*+1}). \tag{11}$$

We have assumed $f_{\hat{\phi},\hat{\tau}}(y_{k^*+1} \mid x_{k^*+1}) > f_{\hat{\phi},\hat{\tau}}(y_{k^*} \mid x_{k^*})$; if not, we increment k^* until this holds.

4 Probability Forecasters for Specific Tasks

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

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

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

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

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

This choice generalizes to $\mathcal{Y} = \mathbb{R}^d$ by having f output the parameters $(\mu, \Sigma) \in \mathcal{Y} \times \mathbb{S}^d_{\succ 0}$ (where $\mathbb{S}^d_{\succ 0}$ is the set of d dimensional symmetric positive definite matrices) of a d dimensional Gaussian distribution. Note that $C_T(x)$ is an ellipsoid $C_T(x) = \mu + \Lambda S^{d-1}$, where $\Lambda \in \mathbb{R}^{d\times d}$ and S^{d-1} is the unit sphere in $\mathbb{R}^d$; we measure the size of $C_T(x)$ as $\|\Lambda\|_F$, where $\|\cdot\|_F$ is the Frobenius norm.

Model-based reinforcement learning. We also consider model-based reinforcement learning, where the goal is to learn a model of the dynamics. In particular, we consider an MDP with states $\mathcal{X}$, actions $\mathcal{U}$, and unknown dynamics $g^* : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$. In general, the goal is to learn a policy that achieves high performance according to some reward function, but in this paper we are interested in obtaining confidence sets on a learned model. Therefore, we assume given a fixed policy

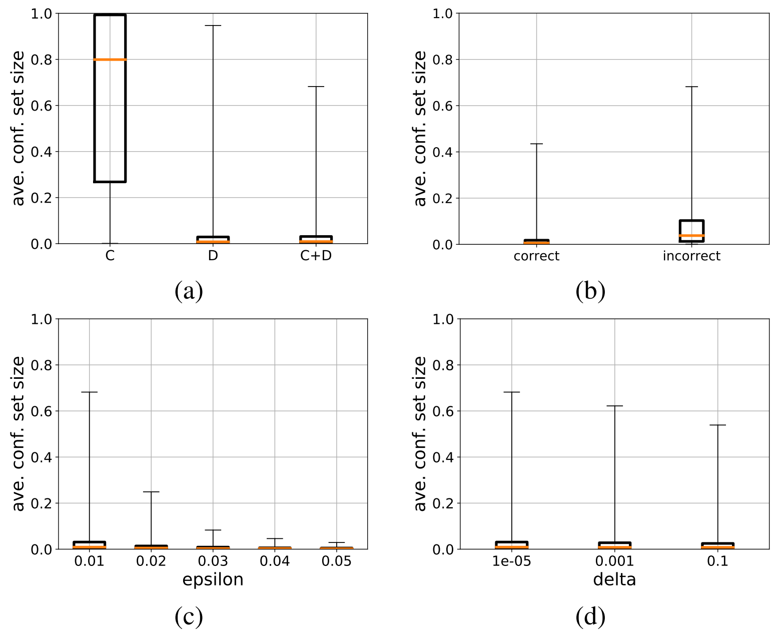


Figure 1: Results on ResNet for ImageNet with $n = 25000$. Default parameters are $\epsilon = 0.01$ and $\delta = 10^{-5}$. We plot the median and min/max confidence set sizes (normalized to 1.0; there are 1000 total labels). (a) Ablation study; C is "calibrated predictor" (i.e., use $f_{\hat{\phi},\hat{\tau}}$ instead of $f_{\hat{\phi}}$), 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 ϵ. (d) Varying δ.

$\pi : \mathcal{X} \to \mathcal{U}$, and let $f^* : \mathcal{X} \to \mathcal{X}$ denote the closed-loop dynamics—i.e., $f^*(x) = g^*(x, \pi(x))$. Additionally, we consider the case where g^* and π may be stochastic, so $f^*(x' \mid x)$ actually encodes a probability distribution over the next state x' given the current state x.

In this context, we assume given a learned model $f(x' \mid x) \approx f^*(x' \mid x)$ that estimates the probability of transitioning from x_t to x_{t+1}; furthermore, we assume that f has the form

$$f(x_{t+1} \mid x_t) = \mathcal{N}(x_{t+1} \mid \mu(x_t), \Sigma(x_t)).$$

Our goal is to obtain confidence sets for our prediction of subsequent states from an initial state x_0. If we only consider one-step predictions $f(x_{t+1} \mid x_t)$, then this setting reduces to the regression setting with labels $\mathcal{Y} = \mathcal{X}$. However, we are interested in the multi-step prediction setting. We use $f_t^*(x' \mid x)$ to denote the true distribution of states after t steps—i.e., $f_1^*(x' \mid x) = f^*(x' \mid x)$ and $f_{t+1}^*(x' \mid x) = \int_{\mathcal{X}} f^*(x' \mid x'') \cdot f_t^*(x'' \mid x) dx''$. We can define the predicted distribution $f_t(x' \mid x)$ of states after t steps in the same way. In principle, the multi-step setting is still a regression problem. The key difference is that it is in general intractable to compute $f_t(x' \mid x)$ in closed form since f can be nonlinear. Instead, we use a heuristic to estimate this distribution. In particular, we use a Gaussian distribution $\tilde{f}_t(x' \mid x) = (\tilde{\mu}(x), \tilde{\Sigma}(x))$, where $\tilde{\mu}(x) = \bar{x}_t$, where $\bar{x}_0 = x$ and $\bar{x}_{t+1} = \mu(\bar{x}_t)$, and

$$\tilde{\Sigma}(x) = \Sigma(\bar{x}_0) + ... + \Sigma(\bar{x}_{t-1}). \tag{12}$$

We use the probability forecaster $\tilde{f}_t$ for calibrating the predicted state distribution after t steps. Then, the problem of predicting confidence sets for $f_t^*(x' \mid x)$ reduces to the case of regression.

5 EXPERIMENTS

ResNet for ImageNet. We use our algorithm to compute confidence sets for ResNet (He et al., 2016) on ImageNet (Russakovsky et al., 2015), for $\epsilon = 0.01$, $\delta = 10^{-5}$, and $n = 25000$ validation images. We show the results in Figure 1. In (a), we compare to two ablations. In particular, C refers to performing an initial temperature scaling step to calibrate the neural network predictor (i.e., using $f_{\hat{\phi}}$ instead of $f_{\hat{\phi},\hat{\tau}}$, and (ii) D refers to using Theorem 1 instead of the VC generalization bound.

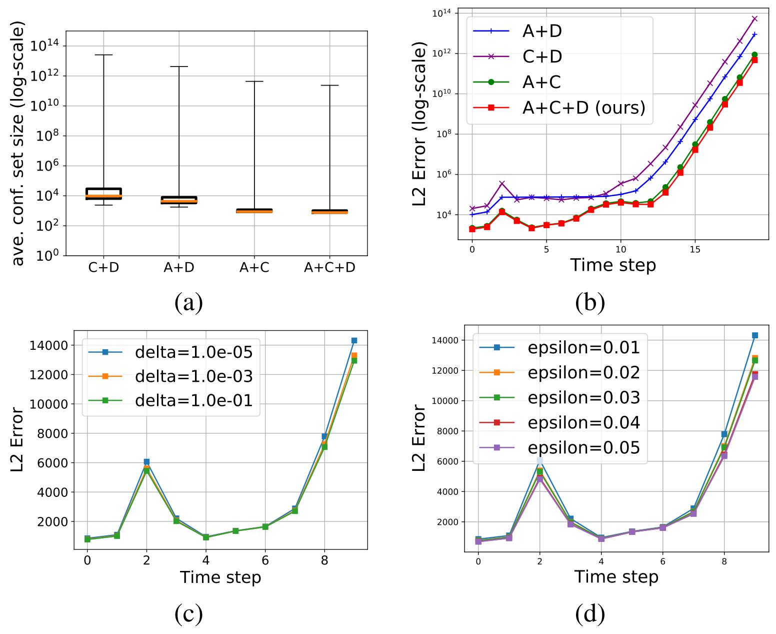


Figure 2: Results on the dynamics model for the half-cheetah with $n = 5000$. Default parameters are $\epsilon = 0.01$ and $\delta = 10^{-5}$. (a) Ablation study; A is "accumulated variance" (i.e., use $\tilde{\Sigma}(x)$ instead of $\Sigma(x)$), and C and D are as for ResNet. We plot the median and min/max confidence set sizes (see Section 4), averaged across $t \in \{0, 1, ..., 9\}$. (b) Same ablations, but with per time step errors. We plot the L_2 error of the confidence set size compared to the true error (i.e., the absolute difference between the predicted and true trajectories). (c) Varying ϵ, and (d) varying δ, for $t \in \{0, 1, ..., 9\}$.

Thus, $C + D$ refers to our approach. As can be seen, using Theorem 1 is performs substantially better than using the VC generalization bound. Using calibrated predictor produces a smaller gain; nevertheless, there is a noticeable reduction in the maximum confidence set size.

In (b), we show the confidence 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 ϵ and δ, respectively. As expected, the dependence on ϵ is much more pronounced (note that δ is log-scale). Finally, we show additional results in Appendix B.

Half-cheetah. We use our algorithm to compute confidence sets for a probabilistic neural network dynamics model (Chua et al., 2018) for the half-cheetah environment (Brockman et al., 2016), for $\epsilon = 0.01$, $\delta = 10^{-5}$, $t = 20$ time steps, and $n = 5000$ validation rollouts. When using temperature scaling to calibrate $f_{\hat{\phi}}$ to obtain $f_{\hat{\phi},\hat{\tau}}$, we calibrate each dimension of the state space independently (i.e., we fit d parameters, where d is state space dimension). We show the results in Figure 2.

In (a), we compare to three ablations for $t = 20$. In addition to C and D (which are as for ResNet), A refers to using the accumulated variance $\tilde{\Sigma}(x_0)$ instead of the one-step predicted variance $\Sigma(x_{t-1})$. Thus, $A+C+D$ refers to our approach. In (b), we show the same ablations over the entire trajectory until $t = 20$. Unlike ResNet, using the calibrated predictor produces a large gain. In contrast, the gain from using Theorem 1 is very small. Using the accumulated confidence produces a large gain.

In (c) and (d), we show how the sizes vary with ϵ and δ, respectively. The trends are similar to the ones we observed for ResNet. Finally, we show additional results in Appendix B.

6 Conclusion

We have proposed an algorithm for constructing PAC confidence sets for deep neural networks. 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).

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.

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.

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. Confidence sets with expected sizes for multiclass classification. *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.

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.

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

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. Classification with confidence. *Biometrika*, 101(4):755–769, 2014.

Jing Lei, Max GSell, 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. In *35th International Conference on Machine Learning, ICML 2018*, volume 9, pp. 6473–6482, 2018.

John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 1999.

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.

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.

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

Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings 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. In *ICLR*, 2017.

A Proof of Theorem 1

Equivalence to binary classification. First, we show that the problem of estimating T is equivalent to a binary classification problem. In particular, define $\phi : \mathcal{Z} \to \mathbb{R}_{>0}$ by $\phi((x,y)) = -\log f(y \mid x)$, where f is the probability forecaster used to construct $\mathcal{C}$. Then, given $T \in \Theta = \mathbb{R}_{>0}$, define $M_T : \mathbb{R}_{>0} \to \{0,1\}$ by $M_T(t) = \mathbb{I}[t \leq T]$. Note that the space of confidence set predictors $\mathcal{C}$ is equivalent to the space $\mathcal{M} = \{M_T \mid T \in \Theta\}$—i.e., for every $T \in \Theta$, we have

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

Furthermore, the problem of estimating a confidence set predictor on a validation set $Z_{\text{val}} \sim D^n$ is equivalent to the problem of estimating $M_T \in \mathcal{M}$ on the validation set

$$\tilde{Z}_{\text{val}} = \{(\phi(z), 1) \mid z \in Z_{\text{val}} \subseteq \tilde{\mathcal{X}} \times \tilde{\mathcal{Y}}\},$$

where $\tilde{\mathcal{X}} = \mathbb{R}_{>0}$ and $\tilde{\mathcal{Y}} = \{0,1\}$. Note that $\tilde{Z}_{\text{val}} \sim \tilde{D}$, where the first component of $\tilde{D}$ is the distribution over $\tilde{\mathcal{X}}$ induced by ϕ from D, and the second component is the distribution over $\tilde{\mathcal{Y}}$ that places all probability mass on 1—i.e.,

$$\tilde{D}((t,1)) = \mathbb{P}_{z \sim D}[\phi(z) = t]$$

In summary, the problem of estimating C_T on a validation set $Z_{\text{val}} \sim D^n$ is equivalent to the problem of estimating M_T on a validation set $\tilde{Z}_{\text{val}} \sim \tilde{D}^n$, which is a binary classification problem with function family $\mathcal{M}$ parameterized by $T \in \Theta$ and loss function

$$\tilde{L}(M) = \mathbb{P}_{(t,a) \sim \tilde{D}}[M(t) \neq a].$$

In particular, note that $L(C_T) = \tilde{L}(M_T)$. Thus, letting $\delta_0 = \sum_{i=0}^{k} \binom{n}{i} \epsilon^i (1-\epsilon)^{n-i}$, a bound

$$\mathbb{P}_{\tilde{Z}_{\text{val}} \sim \tilde{D}^n}\left[\tilde{L}(M_{\hat{T}}) > \epsilon\right] \leq \delta_0 \tag{13}$$

would imply our desired bound $\mathbb{P}_{Z_{\text{val}} \sim D^n}\left[L(C_{\hat{T}}) > \epsilon\right] \leq \delta_0$.

Generalization bound. Let T^* be the smallest T for which $\tilde{L}(M_T) = \epsilon$; then, (13) is equivalently

$$\mathbb{P}_{\tilde{Z}_{\text{val}} \sim \tilde{D}^n}\left[\hat{T} < T^*\right] \leq \delta_0,$$

since $\hat{T} \geq T^*$ implies that $\tilde{L}(M_{\hat{T}}) \leq \tilde{L}(M_{T^*}) = \epsilon$, and vice versa. [1] Consider the event $\hat{T} < T^*$; recalling that $\hat{T}$ must satisfy $\hat{L}(C_{\hat{T}}; Z_{\text{val}}) \leq \alpha$, so on this event, we have

$$k \geq \sum_{(t,a) \in \tilde{Z}_{\text{val}}} \mathbb{I}[M_{\hat{T}}(t) \neq a] = \sum_{(t,a) \in \tilde{Z}_{\text{val}}} \mathbb{I}[t > \hat{T}] \geq \sum_{(t,a) \in \tilde{Z}_{\text{val}}} \mathbb{I}[t > T^*]$$

where $k = n \cdot \alpha(n, \epsilon, \delta)$. Thus, we have

$$\mathbb{P}_{\tilde{Z}_{\text{val}} \sim \tilde{D}^n}\left[\hat{T} < T^*\right] \leq \mathbb{P}_{\tilde{Z}_{\text{val}} \sim \tilde{D}^n}\left[\sum_{(t,a) \in \tilde{Z}_{\text{val}}} \mathbb{I}[t > T^*] \leq k\right].$$

The event in the right-hand side of this inequality is essentially the sum of k i.i.d. random variables $\mathbb{I}[t > T^*] \sim \text{Bernoulli}(\epsilon)$. Thus, this event follows a distribution $\text{Binomial}(n, \epsilon)$, so

$$\mathbb{P}_{\tilde{Z}_{\text{val}} \sim \tilde{D}^n}\left[\hat{T} < T^*\right] \leq \sum_{i=0}^{k} \text{Binomial}(i; n, \epsilon) = \sum_{i=0}^{k} \binom{n}{i} \epsilon^i (1-\epsilon)^{n-i} = \delta_0,$$

as claimed. ■

B Additional Results

Table 2 shows examples of ResNet confidence set sizes for ImageNet images. Table 3 shows results for varying ϵ, δ on ResNet. Tables 4 & 5 show results for varying ϵ, δ on the Half-Cheetah.

[1] We implicitly assume that the distribution $\tilde{D}$ is continuous, but this assumption is very mild.

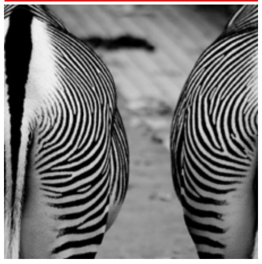
	$\lvert C(x)\rvert = 1$	$5 \le \lvert C(x)\rvert \le 10$	$50 \le \lvert C(x)\rvert \le 100$	$\lvert C(x)\rvert \ge 200$
airship				
zebra				
banana				
carousel				
starfish				
street sign				

Table 2: ImageNet images with varying ResNet confidence set sizes. The confidence set sizes are on the top. The true label is on the left-hand side. Incorrectly labeled images are boxed in red.

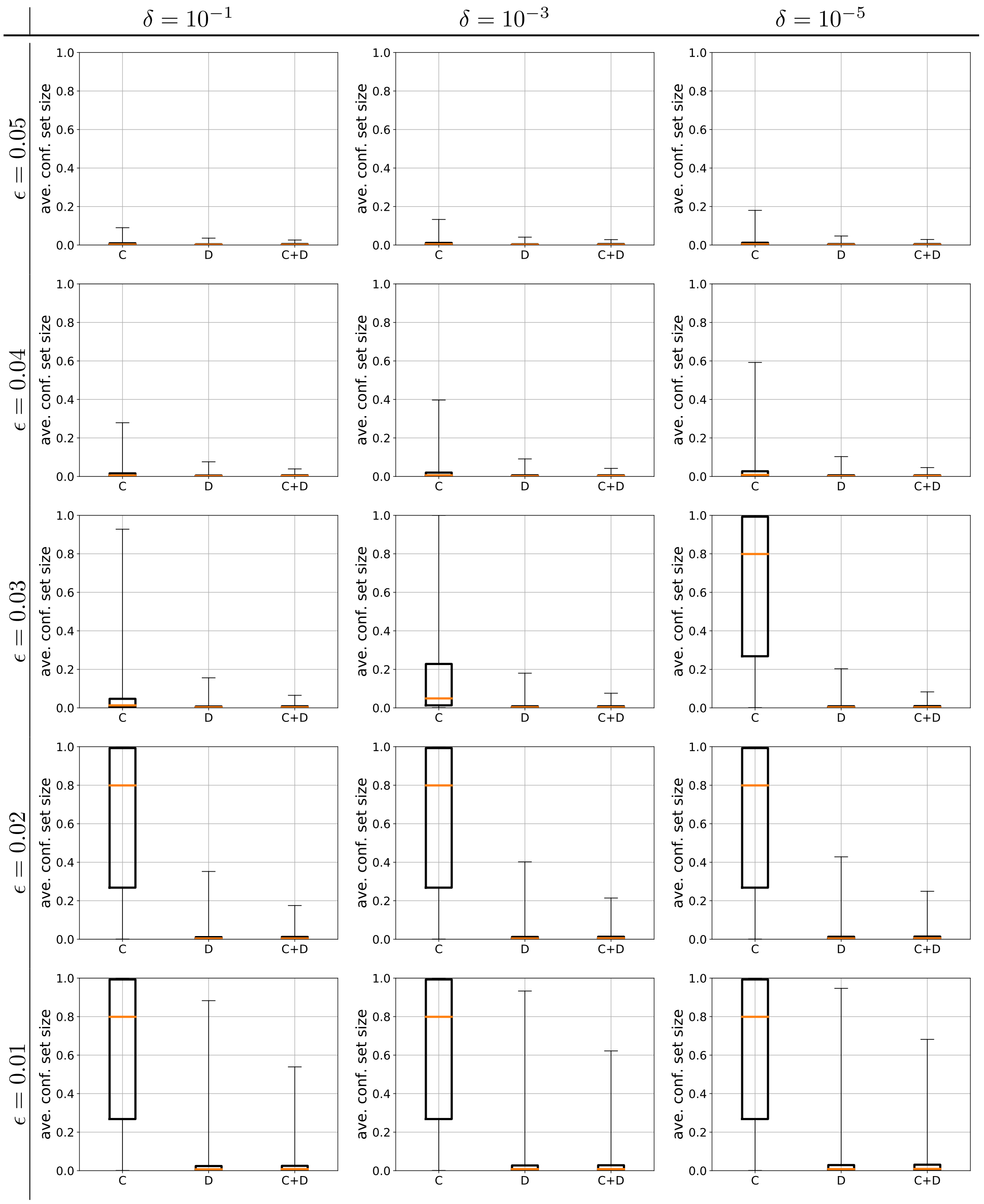


Table 3: Confidence set sizes for ResNet trained on ImageNet, for varying ϵ, δ and for $n = 25000$. The plots are as in Figure 1.

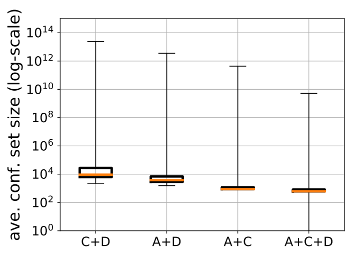
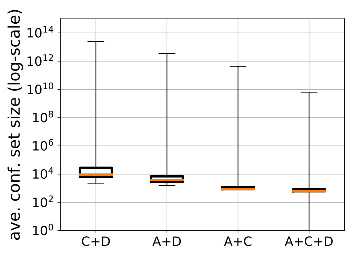
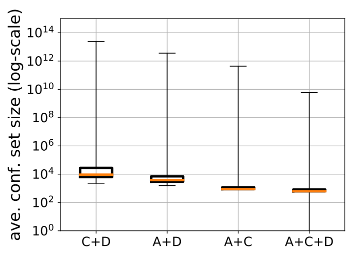
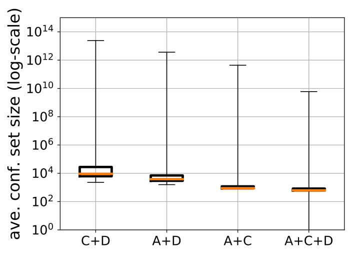
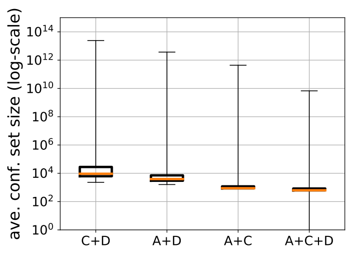
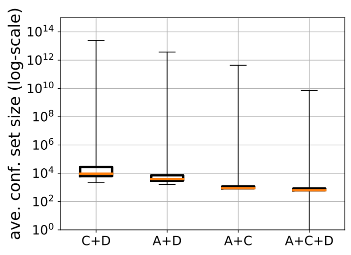
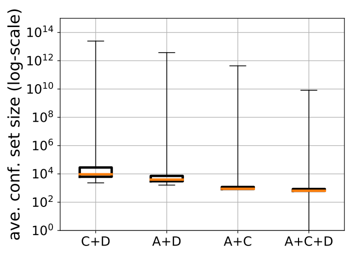
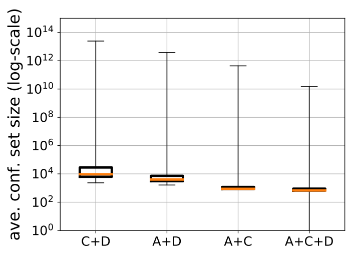
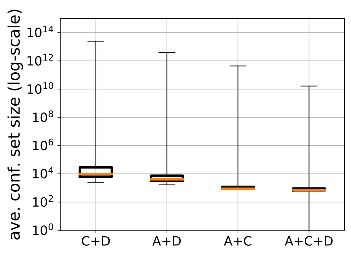
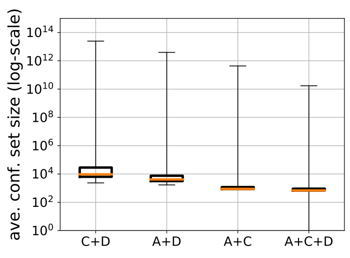
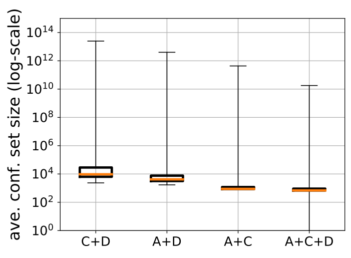
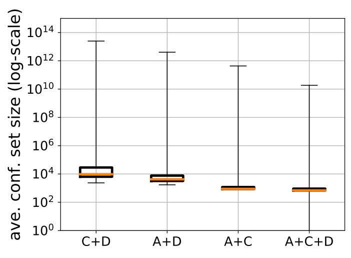
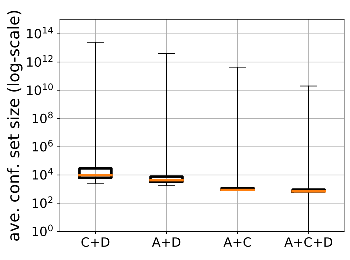
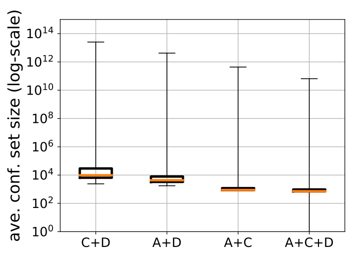
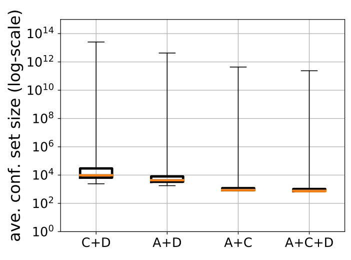
	$\delta = 10^{-1}$	$\delta = 10^{-3}$	$\delta = 10^{-5}$
$\epsilon = 0.05$			
$\epsilon = 0.04$			
$\epsilon = 0.03$			
$\epsilon = 0.02$			
$\epsilon = 0.01$			

Table 4: 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 2 (a).

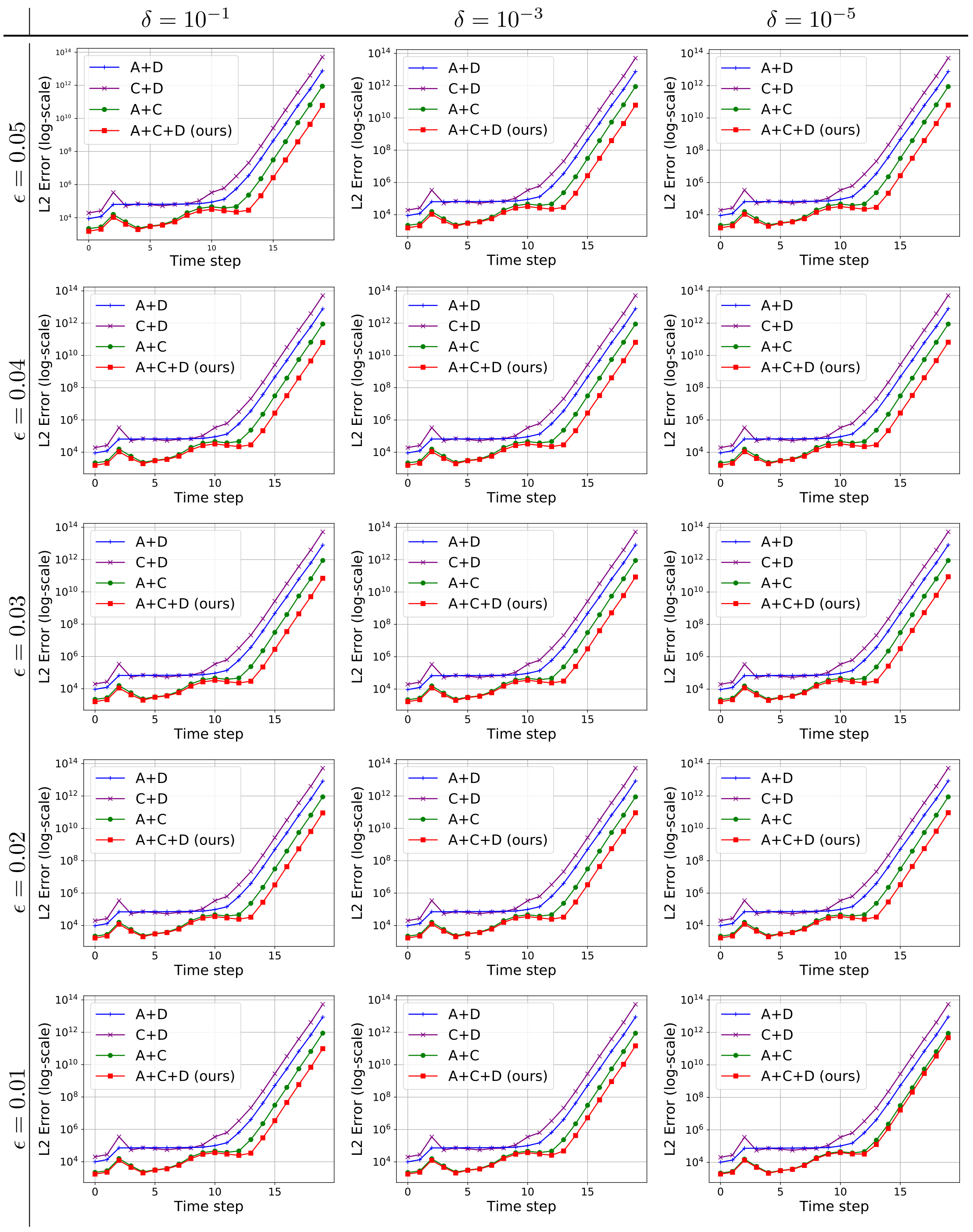


Table 5: Confidence set sizes for a neural network dynamics model trained on the half-cheetah environment, for varying ϵ, δ and for $n = 5000$. The plots are as in Figure 2 (b).