Experts Don't Cheat: Learning What You Don't Know By Predicting Pairs

Daniel D. Johnson [1 2] **Daniel Tarlow** [1] **David Duvenaud** [2] **Chris J. Maddison** [2]

Abstract

Identifying how much a model $\hat{p}^{\theta}_{Y|X}$ knows about the stochastic real-world process $p_{Y|X}$ it was trained on is important to ensure it avoids producing incorrect or "hallucinated" answers or taking unsafe actions. But this is difficult for generative models because probabilistic predictions do not distinguish between per-response noise (aleatoric uncertainty) and lack of knowledge about the process (epistemic uncertainty), and existing epistemic uncertainty quantification techniques tend to be overconfident when the model underfits. We propose a general strategy for teaching a model to both approximate $p_{Y|X}$ and also estimate the remaining gaps between $\hat{p}^{\theta}_{Y|X}$ and $p_{Y|X}$: train it to predict *pairs* of independent responses drawn from the true conditional distribution, allow it to "cheat" by observing one response while predicting the other, then measure how much it cheats. Remarkably, we prove that being good at cheating (i.e. cheating whenever it improves your prediction) is equivalent to being *second-order calibrated*, a principled extension of ordinary calibration that allows us to construct provably-correct frequentist confidence intervals for $p_{Y|X}$ and detect incorrect responses with high probability. We demonstrate empirically that our approach accurately estimates how much models don't know across ambiguous image classification, (synthetic) language modeling, and partially-observable navigation tasks, outperforming existing techniques.

1. Introduction

When a generative model $\hat{p}^{\theta}_{Y|X}$ (such as a large language model) is trained to imitate a stochastic real-world process $p_{Y|X}$, it's important to identify what the model doesn't know about the process. Missing information can cause even well-trained models to "hallucinate" incorrect claims (Ji et al., 2022; Kalai & Vempala, 2023), make unjustified decisions (Hébert-Johnson et al., 2018), or exhibit "self-delusions" that conflate cause and effect (Ortega et al., 2021). Unfortunately, detecting missing information is very difficult when the true responses Y are not deterministic functions of the input X, because probabilistic predictions made by $\hat{p}^{\theta}_{Y|X}$ must account for both the model's uncertainty about the process (called "epistemic uncertainty") and the variability intrinsic to $p_{Y|X}$ ("aleatoric uncertainty"). For example, if responding to a query X = "Tell me about digit 5641 of π", the predicted probability of a response (e.g. "That is 7") may be small either because the model does not know how $p_{Y|X}$ would respond (e.g. whether the answer is actually "That is 4"), or simply because there are many plausible responses under $p_{Y|X}$ (e.g. "Sure, it's an odd number").

If we want to determine whether our model knows enough about $p_{Y|X}$ for us to trust its responses, we cannot rely on the value of $\hat{p}^{\theta}_{Y|X}(y|x)$ alone, since it may just be small be-

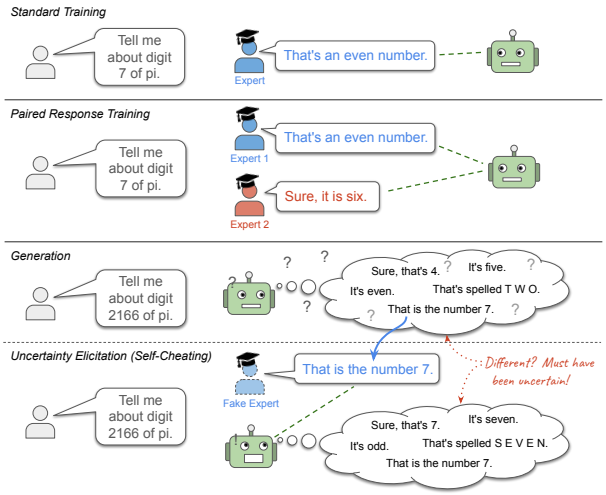

Figure 1. We train a model (green $\hat{p}_{\theta}$) to predict pairs of i.i.d. ground-truth answers (blue ● and red ●), and allow it to "cheat" by observing one (●) while predicting the other (●). Calibrated models only need to cheat when there is something they don't know, so the amount that the model cheats when its own guesses are presented as expert answers can be used to construct provably-correct "cheat-corrected" estimates of how close $\hat{p}^{\theta}_{Y|X}$ is to $p_{Y|X}$.

[1]Google DeepMind [2]University of Toronto, Department of Computer Science, Ontario, Canada. Correspondence to: Daniel D. Johnson <ddjohnson@cs.toronto.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

cause $p_{Y|X}(y|x)$ was small. Instead, what matters is whether $\hat{p}^{\theta}_{Y|X}(y|x)$ is close to $p_{Y|X}(y|x)$. Unfortunately, although we can use metrics like cross-entropy or marginal likelihood to measure improvements in $\hat{p}^{\theta}_{Y|X}$ toward $p_{Y|X}$, we do not generally know the entropy of $p_{Y|X}$ itself, so it is difficult to know how much our model can still improve. In fact, when training on a dataset of (X, Y) pairs it is in general *impossible* to tell how close $\hat{p}^{\theta}_{Y|X}$ is to $p_{Y|X}$ without making assumptions about $p_{Y|X}$ (Barber, 2020). And if we make assumptions that turn out to be false, ensembling or Bayesian-inference-based approaches can produce highly-confident low-uncertainty estimates despite converging to a model that fails to fit important patterns in the data.

In this work, we show that these limitations can be overcome without making assumptions about $p_{Y|X}$ if we instead make a small modification to the training procedure: collect and train on *pairs* of responses (Y_1, Y_2) for each X. Our strategy is based on the following intuition: if an unscrupulous student doesn't know the answer to a question, they could improve their guess by peeking at someone else's answer. By analogy, if a model's prediction $\hat{p}^{\theta}_{Y|X}(\cdot|x)$ does not match the true distribution $p_{Y|X}(\cdot|x)$, the model should be able to improve its prediction if it *cheats* by peeking at a sample $y_1 \sim p_{Y|X}(\cdot|x)$ from the distribution first. And since models only benefit from cheating when they do not already know the distribution, the amount that a calibrated model cheats gives us exactly what we need to robustly estimate the gaps between $\hat{p}^{\theta}_{Y|X}$ and $p_{Y|X}$. Our contributions are:

- We define *second-order calibration*, an extension of ordinary calibration that requires models to additionally report how much the true probabilities $p_{Y|X}(\cdot|x)$ (co)vary around $\hat{p}^{\theta}_{Y|X}(\cdot|x)$ when there are inputs the model cannot distinguish (Figure 2). We also demonstrate that popular epistemic uncertainty quantification approaches are not second-order calibrated under misspecification (Figure 3).
- We show that second-order calibration is equivalent to ordinary calibration over pairs of responses (y_1, y_2), and propose a simple modification to standard maximum-likelihood training ("*training models to cheat*" as in Figure 1) which incentivizes models to become second-order calibrated given sufficient capacity and training data.
- We prove that, given a calibrated model of pairs, you can construct confidence intervals for the true probabilities $p_{Y|X}(y|x)$ and reliable tests for "statistical hallucinations" (responses y with $p_{Y|X}(y|x) = 0$). Our tests rely on a novel and easily-computable *cheat-corrected epistemic confidence* metric, and can be combined with most off-the-shelf decoding strategies to construct new selective decoders with bounded hallucination rates.
- For binary $\mathcal{Y} = \{0, 1\}$, we further show that you can construct nontrivial confidence intervals for $p_{Y|X}$ even with a miscalibrated model as long as you have a calibration set of paired responses, without making any assumptions about the form of $p_{Y|X}$. This means that impossibility results for distribution-free probability regression (Barber, 2020) do not apply when we use paired responses.
- We demonstrate that pair-based variance estimates are empirically second-order well-calibrated on the CIFAR-10H perceptual uncertainty dataset (Peterson et al., 2019), outperforming a variety of existing uncertainty quantification baselines while only requiring small modifications to the data format and output layer.
- We also train Transformer (Vaswani et al., 2017) sequence models on paired responses in synthetic language modeling and partially-observable gridworld tasks, and show that our statistical-hallucination tests enable reliable detection of false statements and unsafe actions despite never observing any such errors during training.

2. Second-Order Calibrated Models Report Where They Know The True Conditional

Let $\mathcal{X}$ be a set of inputs (e.g. prompts or images), and $\mathcal{Y}$ be an arbitrary discrete set of possible responses (such as token sequences or class labels). Suppose we train a model $\hat{p}^{\theta}_{Y|X}$ on a dataset collected from a query distribution $p(X)$ and a ground-truth conditional distribution $p_{Y|X}(Y|X)$, with $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$, and we then use this model to predict the distribution of Y for new $X \sim p(X)$ drawn at inference time. How can we tell if our model $\hat{p}^{\theta}_{Y|X}$ knows enough to match $p_{Y|X}$ for these new queries? Specifically, how can we obtain a reliable estimate of the gap between $\hat{p}^{\theta}_{Y|X}(\cdot|x)$ and $p_{Y|X}(\cdot|x)$?

2.1. Calibrated Models Can Be Far From Perfect

A common way to measure the quality of $\hat{p}^{\theta}_{Y|X}$ is to measure its *calibration*: if we aggregate over inputs X that have the same predicted probability $\hat{p}^{\theta}_{Y|X}(y|X)$, we should hope the true fraction for which $Y = y$ to be about $\hat{p}^{\theta}_{Y|X}(y|X)$.

Definition 2.1. Let $\Delta^{\mathcal{Y}}$ denote the set of probability distributions over the discrete space $\mathcal{Y}$. A predictor $\hat{p}^{\theta}_{Y|X} : \mathcal{X} \to \Delta^{\mathcal{Y}}$ is **(first-order) calibrated** if there exists a **grouping function** $\Phi : \mathcal{X} \to \mathcal{Z}_{\Phi}$ such that $\hat{p}^{\theta}_{Y|X}$ maps each input $x \in \mathcal{X}$ to the average ground-truth distribution $p_{Y|X}$ across random inputs X in the same *equivalence class* $[x]_{\Phi} = \{x' : \Phi(x) = \Phi(x')\} \subset \mathcal{X}$:

$$\begin{aligned} \hat{p}^{\theta}_{Y|X}(y|x) &= \mathbb{E}\big[p_{Y|X}(y|X) \;\big|\; X \in [x]_{\Phi}\big] \\ &= p\big(Y = y \;\big|\; \Phi(X) = \Phi(x)\big). \end{aligned} \tag{1}$$

Calibration is usually defined for the specific grouping function $\Phi^{\theta}_{Y|X}: \mathcal{X} \to \mathbb{R}^{\mathcal{Y}}$ with $\Phi^{\theta}_{Y|X}(x)_y = \hat{p}^{\theta}_{Y|X}(y|x)$, so that the groups are the subsets of $\mathcal{X}$ that map to the same predicted

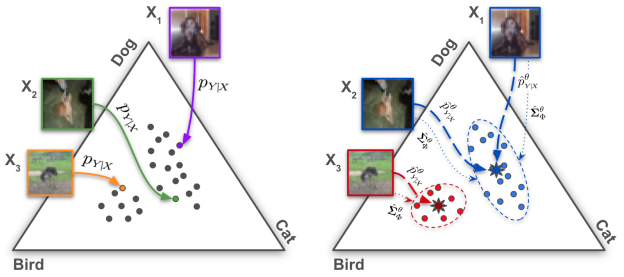

Figure 2. Each input point x (e.g. an ambiguous image) has its own ground-truth response distribution $p_{Y|X}(\cdot|x)$ (e.g. possible human annotator labels for x), but first-order calibration only requires the model's prediction $\hat{p}^{\theta}_{Y|X}$ to be an *average* of $p_{Y|X}$ across an arbitrary grouping of examples (red and blue), which means $\hat{p}^{\theta}_{Y|X}$ can still be far from $p_{Y|X}$ for each individual x. A second-order-calibrated model additionally measures the suboptimality of this approximation by predicting the per-group *covariance* $\hat{\mathbf{\Sigma}}^{\theta}$ of $p_{Y|X}$, but this is challenging because $p_{Y|X}$ itself is never observed.

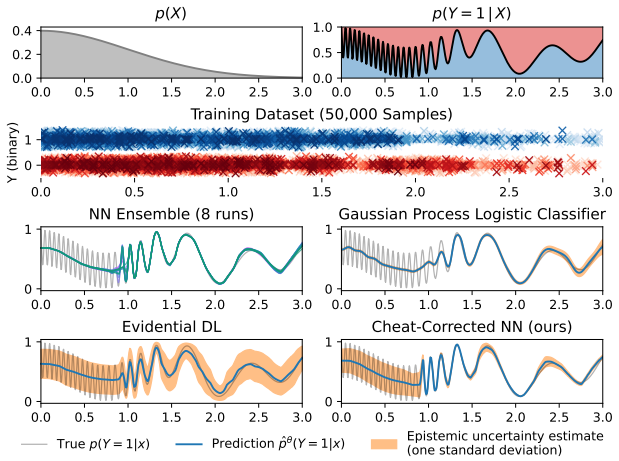

Figure 3. Popular epistemic uncertainty quantification methods are under- or overconfident when $p_{Y|X}$ does not match their assumptions. Given a large number of samples $X \in \mathbb{R}, Y \in \{0, 1\}$, ensembles and misspecified Gaussian process classifiers report low uncertainty at convergence despite failing to match $p_{Y|X}$ around $x \approx 0$; Evidential DL (Sensoy et al., 2018) reports high uncertainty near $x \approx 2.0$ despite fitting well. In contrast, by using *two* samples (Y_1, Y_2) for each X, our method reports uncertainty that matches the true gap $(\hat{p}^{\theta}_{Y|X} - p_{Y|X})^2$ even when it underfits.

distribution (Kumar et al., 2019; Vaicenavicius et al., 2019; Perez-Lebel et al., 2022). We define calibration in terms of an arbitrary grouping function Φ to emphasize that a model $\hat{p}^{\theta}_{Y|X}$ can ignore parts of X and still be well-calibrated; in this case the grouping function $\Phi(x)$ identifies the subsets of $\mathcal{X}$ that the model distinguishes between. These two definitions are equivalent (Gupta et al., 2020), since $\Phi^{\theta}_{Y|X}$ is the coarsest Φ satisfying Equation (1):

Proposition 2.2. *If Eqn. (1) holds for some fixed Φ, then it must also hold for $\Phi^{\theta}_{Y|X} : \mathcal{X} \to \mathbb{R}^{\mathcal{Y}}$, where $\Phi^{\theta}_{Y|X}(x)_y \triangleq \hat{p}^{\theta}_{Y|X}(y|x)$.*

(We defer proofs of all theoretical results to Appendix D.)

A well-calibrated predictor can still be a bad estimate of $p_{Y|X}$ if it fails to distinguish inputs with different true probabilities $p_{Y|X}(y|X)$ and thus averages across them. For example, a calibrated coin-flip predictor might output $\hat{p}^{\theta}_{Y|X}(\text{HEADS}|x) = 50\%$ because it knows coin x is fair, or because it cannot distinguish coins x_+ and x_- with opposite biases. In the first case $\hat{p}^{\theta}_{Y|X}(\text{HEADS}|x) = p_{Y|X}(\text{HEADS}|x)$ and the model is optimal, but in the second the model is suboptimal because it has put inputs with $p_{Y|X}(y|x_+) \neq p_{Y|X}(y|x_-)$ into the same group. This additional error is called the *grouping loss* (Perez-Lebel et al., 2022; Kull & Flach, 2015), which can be lower-bounded but is difficult to upper-bound.

2.2. Second-Order Calibration Measures The Gap

It would be useful if we could get a model to tell us how far $\hat{p}^{\theta}_{Y|X}(y|x)$ might be from $p_{Y|X}(y|x)$ for each x, conditioned on what the model "knows". We make this precise by proposing the following definition.

Definition 2.3. A predictor $\hat{p}^{\theta}_{Y|X} : \mathcal{X} \to \Delta^{\mathcal{Y}}$ and covariance estimator $\hat{\mathbf{\Sigma}}^{\theta} : \mathcal{X} \to \mathbb{R}^{\mathcal{Y} \times \mathcal{Y}}$ are **second-order calibrated** if there exists a grouping function Φ such that $\hat{p}^{\theta}_{Y|X}$ and $\hat{\mathbf{\Sigma}}^{\theta}$ map each input $x \in \mathcal{X}$ to the average *and covariance matrix* of the ground truth probability vector $\boldsymbol{p}_{Y|X}(\cdot|x) \in \Delta^{\mathcal{Y}}$ across inputs X in the same equivalence class under Φ:

$$\hat{p}^{\theta}_{Y|X}(y|x) = \mathbb{E}\big[p_{Y|X}(y|X) \;\big|\; X \in [x]_{\Phi}\big],$$
$$\hat{\mathbf{\Sigma}}^{\theta}(x) = \mathrm{Cov}\Big[\boldsymbol{p}_{Y|X}(\cdot|X), \boldsymbol{p}_{Y|X}(\cdot|X) \;\Big|\; X \in [x]_{\Phi}\Big]$$

where $\boldsymbol{p}_{Y|X}(\cdot|x)_y = p_{Y|X}(y|x)$. We call $\hat{\mathbf{\Sigma}}^{\theta}$ the **epistemic covariance** of the true conditional $\boldsymbol{p}_{Y|X}(\cdot|x)$ under Φ.

If we had a second-order-calibrated predictor, we could use it to identify how tightly concentrated the true probability vector $\boldsymbol{p}_{Y|X}$ is around the model's best guess $\hat{p}^{\theta}_{Y|X}$ (as shown in Figure 2), which would tell us whether $\hat{p}^{\theta}_{Y|X}$ is a good approximation of $p_{Y|X}$. In our coin-flip example, a second-order-calibrated model would report $\hat{\mathbf{\Sigma}}^{\theta}(x)_{y,y} = 0$ if it knows the coin is fair, and $\hat{\mathbf{\Sigma}}^{\theta}(x)_{y,y} > 0$ if it can't tell which way x is biased (i.e. if $\Phi(x) = \Phi(x_+) = \Phi(x_-)$).

Unfortunately, it is not straightforward to construct a second-order-calibrated predictor, because we only observe a *sample* $Y \sim p_{Y|X}(\cdot|x)$ and not the full $\boldsymbol{p}_{Y|X}$. Second-order calibration requires the predictor to distinguish between epistemic and aleatoric uncertainty, but the variance $\mathrm{Var}(Y|\Phi(X))$ of Y itself (for a binary Y) still only measures the total uncertainty and is thus a first-order quantity.

2.3. Existing Epistemic Uncertainty Estimators Under- or Over-estimate The Gap For Underfit Models

Existing techniques for estimating epistemic uncertainty often attempt to estimate how much $p_{Y|X}$ could vary given what the model "knows". For instance, Gaussian processes (Bernardo et al., 1998) and Bayesian neural networks (Goan & Fookes, 2020) impose a prior distribution over the generative process, then evaluate the variance of the prediction under an approximate posterior (Kendall & Gal, 2017). Other related strategies include ensembling (Lakshminarayanan et al., 2016), injecting noise into the model or training process (Gal & Ghahramani, 2015; Osband et al., 2021; Maddox et al., 2019), or predicting a "distribution over distributions" (Sensoy et al., 2018; Malinin & Gales, 2018).

We might hope that these estimates would be second-order calibrated, but unfortunately this is not generally the case, especially if the model is misspecified or underfit relative to $p_{Y|X}$. We demonstrate this in Figure 3 by applying a variety of methods to a fixed $p_{Y|X}$ with both low- and high-frequency variation (discussed more in Appendix F.1). With a large training set, an ensemble and a Gaussian Process classifier both converge to highly confident but incorrect solutions, because the prior was misspecified and did not include $p_{Y|X}$. Evidential DL (Sensoy et al., 2018), on the other hand, is underconfident because its objective biases its uncertainty estimates (Bengs et al., 2022; 2023). In practice, even the largest models are likely to underfit in some regions of $\mathcal{X}$, making this a serious concern if we wish to reliably estimate how far $\hat{p}^{\theta}_{Y|X}$ actually is from $p_{Y|X}$.

3. Second-Order Calibration From Paired Ys

How can we obtain a second-order calibrated model? We now show that making second-order-calibrated predictions about individual response *probabilities* is equivalent to making first-order-calibrated predictions about *paired responses*.

Suppose we have a model $\hat{p}^{\theta}_{Y_1,Y_2|X}(Y_1, Y_2|X)$ predicting a distribution over $\mathcal{Y} \times \mathcal{Y}$, and let $\hat{p}^{\theta}_{Y_1|X}, \hat{p}^{\theta}_{Y_2|X}, \hat{p}^{\theta}_{Y_2|Y_1,X}$ be the induced marginal and conditional distributions. If $\hat{p}^{\theta}_{Y_1,Y_2|X}$ is calibrated at predicting a pair of independent responses $Y_1, Y_2 \overset{iid}{\sim} p_{Y|X}(\cdot\,|X)$, it must be the case that

$$\hat{p}^{\theta}_{Y_1,Y_2|X}(y_1, y_2|x) = \mathbb{E}\Big[p_{Y|X}(y_1|X) \cdot p_{Y|X}(y_2|X) \,\Big|\, X \in [x]_{\Phi}\Big]$$

for some Φ. How much should we expect y_2 to depend on y_1 according to this model? Although Y_1 and Y_2 are independent given X, they may not be independent conditioned on $\Phi(X)$, i.e. conditioned on what our model "knows" about X. In this case, we should expect a calibrated model to "*cheat*" by using information about y_1 to better inform its prediction of y_2. We can quantify this by measuring how correlated the possible outcomes are under the model:

Definition 3.1. The **pair covariance** of $\hat{p}^{\theta}_{Y_1,Y_2|X}$ is

$$\hat{\mathbf{\Sigma}}^{\theta}_{Y_1,Y_2|X}(x)_{y_i,y_j} \triangleq \hat{p}^{\theta}_{Y_1,Y_2|X}(y_i, y_j|x) - \hat{p}^{\theta}_{Y_1|X}(y_i|x)\, \hat{p}^{\theta}_{Y_2|X}(y_j|x)$$

$\hat{\mathbf{\Sigma}}^{\theta}_{Y_1,Y_2|X}(x)_{y_i,y_j}$ is the difference between the predicted joint and what we would expect if Y_1 and Y_2 were independent given $\Phi(X)$. It turns out that this is exactly what we need to construct a second-order-calibrated predictor of $p_{Y|X}$:

Theorem 3.2. *If $\hat{p}^{\theta}_{Y_1,Y_2|X}$ is first-order calibrated at predicting pairs (Y_1, Y_2), then its marginal $\hat{p}^{\theta}_{Y_1|X}$ and pair covariance $\hat{\mathbf{\Sigma}}^{\theta}_{Y_1,Y_2|X}$ are second-order calibrated at predicting $p_{Y|X}$. Moreover, this is a bijection: for any second-order-calibrated $(\hat{p}^{\theta'}_{Y|X}, \hat{\mathbf{\Sigma}}^{\theta'})$, there is a unique first-order-calibrated $\hat{p}^{\theta}_{Y_1,Y_2|X}$ with $\hat{p}^{\theta'}_{Y|X} = \hat{p}^{\theta}_{Y_1|X}$ and $\hat{\mathbf{\Sigma}}^{\theta'} = \hat{\mathbf{\Sigma}}^{\theta}_{Y_1,Y_2|X}$.*

This equivalence means that techniques for training first-order-calibrated models can also be used to construct second-order calibrated models whenever it is possible to draw multiple samples from $p_{Y|X}$ (e.g. by asking two random human experts to label X). In particular, we propose to directly train a model $\hat{p}^{\theta}_{Y_1,Y_2|X}(Y_1, Y_2|X)$ to predict paired responses by minimizing the standard cross-entropy loss

$$-\mathbb{E}_{\substack{X\sim p(X),\\ Y_1,Y_2\sim p_{Y|X}}}\Big[\log \hat{p}^{\theta}_{Y_1,Y_2|X}(Y_1, Y_2|X)\Big]$$

over a dataset of $(X^{(i)}, Y_1^{(i)}, Y_2^{(i)})$ triples. Since cross-entropy is a proper scoring rule (Kull & Flach, 2015), we can expect that our model will become more calibrated over $\mathcal{Y} \times \mathcal{Y}$ as it improves. Indeed, calibration is linked to generalization ability (Carrell et al., 2022) and hallucination behavior (Kalai & Vempala, 2023) and tends to emerge in sufficiently-high-capacity models (Błasiok et al., 2023; OpenAI, 2023; Kadavath et al., 2022). We note that if our model is explicitly factorized as

$$\hat{p}^{\theta}_{Y_1,Y_2|X}(y_1, y_2|x) = \hat{p}^{\theta}_{Y_1|X}(y_1|x) \cdot \hat{p}^{\theta}_{Y_2|Y_1,X}(y_2|y_1, x)$$

(e.g. an autoregressive model), we expect it to learn to "cheat" by copying information from Y_1 to Y_2 whenever there are regularities between Y_1 and Y_2 that aren't explained away by what the model knows. This is exactly what we want, because calibration *requires* $\hat{p}^{\theta}_{Y_1,Y_2|X}$ to cheat whenever $\hat{p}^{\theta}_{Y_1|X} \neq p_{Y|X}$; we can then use Theorem 3.2 to determine how close $\hat{p}^{\theta}_{Y_1|X}$ is to $p_{Y|X}$. Informally, an expert doesn't need to cheat, so if you let your model cheat and it does, it must not know the answer to your question.

4. Bounding Approximation Error With Pairs

4.1. Pair Predictors Can Bound Their Own Individual-Response Errors By Self-Cheating

We now derive a number of properties which are particularly useful when using $\hat{p}^{\theta}_{Y_1|X}$ to imitate $p_{Y|X}$: bounded deviation

between $\hat{p}^\theta_{Y|X}$ and $p_{Y|X}$, and bounded probability of producing outputs where $p_{Y|X}(y|x) = 0$. These results rely on the fact that, conditioned on the matrix $\Phi^\theta_{Y_1,Y_2|X}(x) \in \mathbb{R}^{\mathcal{Y}\times\mathcal{Y}}$ of model outputs (with $\Phi^\theta_{Y_1,Y_2|X}(x)_{y_1,y_2} = \hat{p}^\theta_{Y_1,Y_2|X}(y_1, y_2|x)$), we can treat $p_{Y|X}(y|X)$ as a random variable whose mean is $\hat{p}^\theta_{Y_1|X}(y|X)$ and variance is $\hat{V}^\theta_{\text{CHEAT}}(y|X)$, defined below:

Definition 4.1. The **cheat-corrected epistemic variance** of $p_{Y|X}$ for response y to query x (under $\hat{p}^\theta_{Y_1,Y_2|X}$) is

$$\hat{V}^\theta_{\text{CHEAT}}(y|x) \triangleq \hat{p}^\theta_{Y_1|X}(y|x)\left(\hat{p}^\theta_{Y_2|Y_1,X}(y|y,x) - \hat{p}^\theta_{Y_1|X}(y|x)\right).$$

$\hat{V}^\theta_{\text{CHEAT}}$ can be computed easily by scoring y twice, once under the marginal distribution of Y_1 and once when the model "self-cheats" by conditioning on y (as Y_1) when predicting y again (as Y_2). Furthermore, it agrees with the diagonal entries of $\hat{\mathbf{\Sigma}}^\theta_{Y_1,Y_2|X}(x)$ as long as $\hat{p}^\theta_{Y_1,Y_2|X}$ is symmetric (which is true if $\hat{p}^\theta_{Y_1,Y_2|X}$ is calibrated). This means we can use it to bound the distance between $\hat{p}^\theta_{Y_1|X}$ and $p_{Y|X}$.

Theorem 4.2. *Suppose $\hat{p}^\theta_{Y_1,Y_2|X}$ is calibrated. Let A be any event and $\tilde{Y} \in \mathcal{Y}$ be any (possibly random) value such that $\tilde{Y}, A \perp\!\!\!\perp X \mid \Phi^\theta_{Y_1,Y_2|X}(X)$. Then*

$$\mathbb{E}\left[\left(\hat{p}^\theta_{Y_1|X}(\tilde{Y}|X) - p_{Y|X}(\tilde{Y}|X)\right)^2 \,\middle|\, A\right] = \mathbb{E}\left[\hat{V}^\theta_{\text{CHEAT}}(\tilde{Y}|X) \,\middle|\, A\right].$$

Furthermore, for any $\beta \in (0, 1)$,

$$P\left[\left|\hat{p}^\theta_{Y_1|X}(\tilde{Y}|X) - p_{Y|X}(\tilde{Y}|X)\right| \geq \sqrt{\frac{\hat{V}^\theta_{\text{CHEAT}}(\tilde{Y}|X)}{\beta}} \,\middle|\, A\right] \leq \beta.$$

This is a input-dependent (frequentist) confidence interval for $\tilde{Y}$; if our model reports a small value of $\hat{V}^\theta_{\text{CHEAT}}(\tilde{Y}|X)$, we can guess that $\hat{p}^\theta_{Y_1|X}(\tilde{Y}|X)$ is close to $p_{Y|X}(\tilde{Y}|X)$ and be right most of the time. (For instance, if A is the event where our example coin-flip predictor predicts 50% HEADS with epistemic variance $\leq \epsilon$, at least 95% of the coins with that property must have a bias within $\sqrt{\varepsilon/.05}$ of 50%.)

When $\mathcal{Y}$ is large, we may be less interested in directly estimating $p_{Y|X}$ for a particular y, and more interested in making sure we don't generate any response y for which $p_{Y|X}(y|x)$ was actually zero; we call such a response a *statistical hallucination*.[1] We can do so using the following metric:

Definition 4.3. The **cheat-corrected epistemic confidence** of $\hat{p}^\theta_{Y_1,Y_2|X}$ about response y to query x is

$$C^\theta_{\text{CHEAT}}(y|x) \triangleq \frac{\hat{p}^\theta_{Y_1|X}(y|x)}{\hat{p}^\theta_{Y_2|Y_1,X}(y|y,x)} \quad \text{(or 0 if } \hat{p}^\theta_{Y_1|X}(y|x) = 0\text{)}.$$

[1]The term "hallucination" is often used to mean "output with false factual claims". These count as statistical hallucinations as long as $p_{Y|X}$ never produces them, but statistical hallucinations also include behavior such as taking unsafe actions that $p_{Y|X}$ would avoid, making a math error where $p_{Y|X}$ would be correct, or failing to satisfy any other property of all samples generated by $p_{Y|X}$.

C^θ_{CHEAT} measures the *relative* likelihood with and without self-cheating, with the denominator correcting for the "aleatoric" aspects of y that remain unpredictable even when the model cheats. Similar to $\hat{V}^\theta_{\text{CHEAT}}$, it can be computed easily by scoring y twice. C^θ_{CHEAT} is also properly normalized:

Proposition 4.4. *If $\hat{p}^\theta_{Y_1,Y_2|X}$ is calibrated, then for any $x \in \mathcal{X}, y \in \mathcal{Y}$ we have $0 \leq C^\theta_{\text{CHEAT}}(y|x) \leq 1$, with $C^\theta_{\text{CHEAT}}(y|x) = 1$ if and only if $\hat{p}^\theta_{Y_1|X}(y|x) = p_{Y|X}(y|x)$.*

And we can use it to bound the statistical-hallucination rate of any well-behaved decoding algorithm:

Theorem 4.5. *Suppose $\hat{p}^\theta_{Y_1,Y_2|X}$ is calibrated. Let A be the event that a decoding algorithm responds to a query X, and $\tilde{Y} \in \mathcal{Y}$ be its response. If $A, \tilde{Y} \perp\!\!\!\perp X \mid \Phi^\theta_{Y_1,Y_2|X}(X)$, then the statistical hallucination rate of the generated responses is bounded above as*

$$P\left[p_{Y|X}(\tilde{Y}|X) = 0 \,\middle|\, A\right] \leq 1 - \mathbb{E}\left[C^\theta_{\text{CHEAT}}(\tilde{Y}|X) \,\middle|\, A\right].$$

We can use any decoding strategy that only depends on X through $\hat{p}^\theta_{Y_1,Y_2|X}$, including temperature sampling, top-k/top-p sampling, or beam search (see Zarrieß et al. (2021) for an overview). Moreover, we are free to use $C^\theta_{\text{CHEAT}}(\tilde{Y}|X)$ in the algorithm to ensure that $1 - C^\theta_{\text{CHEAT}}$ is low. For example, these decoding strategies will all have a statistical hallucination rate at most β when $\hat{p}^\theta_{Y_1,Y_2|X}$ is calibrated:

- **Cheat-corrected selective generation / filtering**: Generate $\tilde{Y}$ using an arbitrary off-the-shelf sampler, but reject it (and don't respond) if $1 - C^\theta_{\text{CHEAT}}(\tilde{Y}|X) > \beta$.
- **Cheat-corrected rejection sampling**: Repeatedly sample $\tilde{Y} \sim \hat{p}^\theta_{Y_1|X}$ until $1 - C^\theta_{\text{CHEAT}}(\tilde{Y}|X) < \beta$.
- **Cheat-corrected top-1 search**: Deterministically output (or approximate) $\arg\max_{y\in S} \hat{p}^\theta_{Y_1|X}(y|X)$, where $S = \{y : 1 - C^\theta_{\text{CHEAT}}(\tilde{Y}|X) < \beta\}$, or abstain if $S = \varnothing$.

Selectively responding only when we find a $\tilde{Y}$ with $1 - C^\theta_{\text{CHEAT}}(\tilde{Y}|X) < \beta$ ensures that, conditioned on responding (e.g. on the event A), our responses will be non-hallucinated with probability at least $1 - \beta$.

4.2. Paired Data Enables Distribution-Free Frequentist Confidence Intervals for $p(Y|X)$

Finally, we show that we can adjust imperfectly-calibrated estimators $\hat{p}^\theta_{Y|X} : \mathcal{X} \to \Delta^{\mathcal{Y}}$ and $\hat{V}^\theta : \mathcal{X} \to \mathbb{R}^{\mathcal{Y}}$ to obtain robust statistical guarantees about the unobserved true conditional probabilities $p_{Y|X}(Y|X)$ without assumptions about $p_{Y|X}$, as long as we have access to a held-out *calibration set* $\{(x^{(i)}, y_1^{(i)}, y_2^{(i)})\}_{i=1}^N$ containing paired response data. This demonstrates that the impossiblity result of Barber (2020) does not apply when we have access to two Ys for each X. For simplicity we assume $\mathcal{Y} = \{0, 1\}$.

Algorithm 1 Conservative adjustment of $\hat{V}^\theta$

Input: Calibration set $\{(x^{(i)}, y_1^{(i)}, y_2^{(i)})\}_{i=1}^N$, variance cutoff $\varepsilon > 0$, tolerance α, $\hat{p}^\theta_{Y|X}$, $\hat{V}^\theta$
for $i = 1$ **to** N **do**
  $\hat{p}^{(i)} := \hat{p}^\theta_{Y|X}(1|x^{(i)})$, $\hat{v}_\varepsilon^{(i)} := \max\{\hat{V}^\theta(1|x^{(i)}), \varepsilon\}$
  $s_\varepsilon^{(i)} := (y_1^{(i)} - \hat{p}^{(i)})(y_2^{(i)} - \hat{p}^{(i)})/\hat{v}_\varepsilon^{(i)}$
end for
$(\gamma_\varepsilon^-, \gamma_\varepsilon^+) := \textsc{MeanConfItvl}\big(\{s_\varepsilon^{(i)}\}_{i=1}^N, -\frac{1}{\varepsilon}, \frac{1}{\varepsilon}, \alpha\big)$
return γ_ε^+

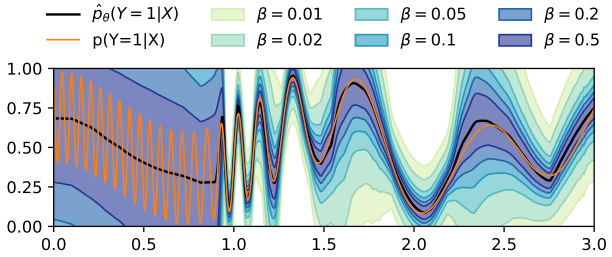

Figure 4. Applying Algorithm 1 to our model from Figure 3 produces frequentist confidence intervals for $p_{Y|X}(y|X)$ which are provably correct with high probability over random X. Here $N = 10^6, \varepsilon = 0.02^2$, and $\alpha = 0.05$; see Appendix B.

Theorem 4.6. *Let $\hat{p}^\theta_{Y|X}$, $\hat{V}^\theta$, and $p_{Y|X}$ be arbitrary. With probability at least $1-\alpha$ (over draws of the calibration set), Algorithm 1 returns a value γ_ε^+ such that, for a randomly sampled input $X \sim p(X)$, and any $\beta \in (0,1), y \in \{0,1\}$,*

$$P\left[\left|\hat{p}^\theta_{Y|X}(y|X) - p_{Y|X}(y|X)\right| \geq \sqrt{\frac{\gamma_\varepsilon^+ \max\{\hat{V}^\theta(y|X), \varepsilon\}}{\beta}}\right] \leq \beta.$$

In Algorithm 1, MeanConfItvl can be any subroutine that builds a $(1-\alpha)$ confidence interval for the mean of a bounded random variable, e.g. Hoeffding's inequality (Hoeffding, 1994) or betting-based algorithms (Waudby-Smith & Ramdas, 2020). Smaller ε allows more precise bounds but requires a well-calibrated $\hat{V}^\theta$ and a large calibration set, and if $\hat{p}^\theta_{Y|X}$ and $\hat{V}^\theta$ are in fact second-order calibrated then γ_ε^+ will approach 1 as $N \to \infty$ and $\varepsilon \to 0$. The failure probability β should be interpreted as an aggregate over $X \sim p(X)$ rather than pointwise; for a fixed process $p_{Y|X}$ and fixed x either $p_{Y|X}(y|x)$ lies in the interval or it does not. We show an example of the resulting confidence intervals in Figure 4, and discuss them further in Appendices B and D.4.

5. Related Work

Decomposing uncertainty with paired Ys. Focusing on regression tasks and asymptotic optimality, Lahlou et al. (2021) estimate aleatoric uncertainty by predicting $(y_1 - y_2)^2$ for two real-valued samples from $p(Y|X)$, then use it to quantify epistemic uncertainty. For classification, Narimatsu et al. (2023) use annotator agreement to quantify aleatoric uncertainty at the population level. Repeated annotations have also been used to improve and evaluate classifiers (Peterson et al., 2019; Schmarje et al., 2022).

Uncertainty via LLM postprocessing. For language models, proposed techniques include verifying, critiquing, or classifying samples (Cobbe et al., 2021; Ni et al., 2023; Li et al., 2022b; Kadavath et al., 2022), or clustering semantically-equivalent samples (Kuhn et al., 2022; Li et al., 2022a; Wang et al., 2022; Chen et al., 2023). This generally requires a task-specific correctness or similarity metric, and may be less applicable for generation tasks without well-defined correct answers. Additionally, most multiple-sample approaches focus on comparing many Ys at inference time, whereas our strategy only uses paired Ys at training time and then scores each Y individually.

Other uses of paired Ys. In other contexts, paired inputs have been used to learn representations (Bromley et al., 1993; Chen et al., 2020), and pairwise losses have been used to train energy-based models (Gutmann & Hyvärinen, 2010). Lin et al. (2018) train a GAN discriminator to distinguish pairs of real v.s. generated images and show that this reduces mode collapse.

Uncertainty via dependence on additional information. Durasov et al. (2022) train a model to predict the same output both with and without feeding in the correct output as an extra input, and use the change in prediction to measure uncertainty. A key difference between this and our cheat-correction procedure is that Durasov et al. treat the output as deterministic (no aleatoric uncertainty) and rely on inductive biases of the predictor rather than calibration. Collier et al. (2022) provide additional privileged information about the label process in order to explain away label noise and improve robustness.

Uncertainty via extensions of calibration. To better measure uncertainty for calibrated models, Perez-Lebel et al. (2022) propose bounding the population grouping error by partitioning the model's feature space. Hébert-Johnson et al. (2018) study *multicalibration*, which requires calibration to hold across all computable subsets of a population.

Distribution-free uncertainty quantification. A number of approaches have been explored for quantifying uncertainty without making assumptions about the functional form of $p(Y|X)$, generally by using a held-out calibration set. Many build on conformal prediction, and use exchangeability to construct high-probability prediction sets; see Angelopoulos & Bates (2021) for an introduction. Related approaches can be used to construct calibrated classifiers (Kumar et al., 2019; Gupta et al., 2020; Park et al., 2020) and randomized predictive distributions (Vovk et al., 2017). We discuss these connections in more detail in Appendix C.

Table 1. **Cheat-corrected uncertainty estimates are better second-order-calibrated than other techniques, while maintaining similar accuracy.** Our primary metrics: *ECE-2* is second-order calibration error of the variance estimate (best ECE-2 in bold), $\mathbb{E}[\hat{v}^\theta]$ is predicted epistemic variance, and $\mathbb{E}[(\hat{p}^\theta - p)^2]$ is actual grouping error (ideally close to $\mathbb{E}[\hat{v}^\theta]$). For comparison, *ECE-1* is first-order calibration error of predicted probabilities, *Acc* is top-1 accuracy on the original labels from CIFAR-10, and *KL* measures the divergence from $p_{Y|X}$ (ground-truth annotator labels) to $\hat{p}^\theta_{Y|X}$. All metrics are averaged over eight random training seeds, and metrics other than Acc and KL are summed across classes.

	CIFAR-10H						w/ Extra Classes, Scrambled				
Method	**ECE-2**	$\mathbb{E}[\hat{v}^\theta]$	$\mathbb{E}[(\hat{p}^\theta - p)^2]$	ECE-1	Acc	KL	**ECE-2**	$\mathbb{E}[\hat{v}^\theta]$	$\mathbb{E}[(\hat{p}^\theta - p)^2]$	ECE-1	KL
Naive NN	0.076	0.142	0.065	0.02	93.9	0.18	0.521	0.682	0.161	0.07	0.71
NN Ensemble	0.039	0.014	0.053	0.03	94.9	0.15	0.134	0.014	0.148	0.03	0.65
Evidential DL	0.377	0.053	0.430	1.04	88.5	1.09	0.387	0.031	0.418	0.79	2.36
SNGP Cov.	0.048	0.005	0.052	0.02	94.9	0.15	0.112	0.033	0.145	0.06	0.63
Epinet	0.056	0.015	0.071	0.02	93.4	0.19	0.089	0.087	0.163	0.07	0.71
Cheat NN	0.018	0.052	0.068	0.03	93.6	0.18	0.022	0.134	0.154	0.07	0.67
Cheat SNGP	**0.009**	0.054	0.052	0.02	94.9	0.15	**0.011**	0.153	0.150	0.04	0.65

Predicting distributions-over-distributions. Some previous techniques (e.g. Sensoy et al., 2018; Malinin & Gales, 2018) have explored measuring uncertainty by predicting a distribution over possible output distributions (sometimes called second-order distributions). However, Bengs et al. (2022; 2023) proved that many such approaches do not incentivize faithful reports of uncertainty. Sale et al. (2023) formalize uncertainty measures for second-order distribution predictors in terms of distances to sets of reference distributions. Note that our work uses "second-order" in the sense of the second-moment statistics in Theorem 3.2, not second-order distributions. Our approach does not predict a full distribution over distributions.

6. Experiments

6.1. Classifying Ambiguous Images

We demonstrate our technique on CIFAR-10H (Peterson et al., 2019), a relabeling of the CIFAR-10 test set (Krizhevsky, 2009) by > 50 independent annotators per image. We cast it as a distribution-matching problem rather than an accuracy-maximization problem: the goal is to estimate the fraction of human annotators assigning each label y to each image x. In this setting, we expect epistemic uncertainty quantification techniques to distinguish between between images that *human annotators* find ambiguous and images that the *model* has not learned to identify. Our primary evaluation metric is second-order expected calibration error (ECE-2), the difference between each technique's variance estimate $\hat{V}^\theta$ and the true squared error $\big(\hat{p}_\theta(Y|X) - p(Y|X)\big)^2$, on an in-distribution test set. Since some uncertainty-quantification methods may affect predictive accuracy, we additionally report the ordinary expected calibration error of $\hat{p}_\theta(Y|X)$ relative to the true annotator labels (ECE-1), the KL divergence between $\hat{p}_\theta(Y|X)$ and the empirical annotator distribution, and the top-1 accuracy with respect to the clean CIFAR-10 labels. We compute ECE-1 and ECE-2 by averaging over 100 quantile bins and summing across classes, as described in Appendix F.2.

We train pair-prediction models $\hat{p}^\theta_{Y_1,Y_2|X}$ to jointly predict two random annotator labels for each minibatch example, with a symmetric 10×10 softmax output head and either an ordinary wide ResNet backbone (**Cheat NN**) from Zagoruyko & Komodakis (2016) or a SNGP backbone (**Cheat SNGP**) as proposed by Liu et al. (2020). We then use the marginal $\hat{p}^\theta_{Y_1|X}$ and cheat-corrected variance $\hat{V}^\theta_{\text{CHEAT}}$ for evaluation. We observed that our models occasionally overfit on the small dataset and produced negative $\hat{V}^\theta_{\text{CHEAT}}$ estimates due to miscalibration; we regularize them by adding a small penalty for negative eigenvalues, since $\hat{p}^\theta_{Y_1,Y_2|X}(\cdot,\cdot|x)$ must be positive semidefinite if $\hat{p}^\theta_{Y_1,Y_2|X}$ is calibrated (proven in Appendix E).

We compare our approach to a variety of existing uncertainty quantification techniques: **SNGP Cov.** (Liu et al., 2020), which uses spectral normalization and a Laplace random-features approximation to a Gaussian process covariance; **Evidential DL** (Sensoy et al., 2018), which uses a regularized Dirichlet output to estimate epistemic uncertainty; **Epinet** (Osband et al., 2021), which models uncertainty by feeding a random "index" input through a fixed "prior" network and a learned corrector; **NN Ensemble** (Lakshminarayanan et al., 2016), which uses the mean and variance across 8 independent ResNet models; and **Naive NN**, which uses $\hat{p}_\theta(Y|X)(1 - \hat{p}_\theta(Y|X))$ as an estimate of variance (i.e. assuming Y is a deterministic function of X). We train these baselines by trating the two randomly-selected annotator labels as separate minibatch examples.

Since CIFAR-10H includes only the original CIFAR-10 test set, we pretrain all models on CIFAR-10N (Wei et al., 2021), a relabeling of CIFAR-10's training set by three annotators per image. We then divide CIFAR-10H's images

into two disjoint 5,000-image subsets (with > 50 annotator labels per image), using the first to train/validate and the second for evaluation metrics. We tune hyperparameters to maximize likelihood on our validation set, but intentionally avoid tuning based on second-order calibration since this may not be computable under a standard training setup.

As shown in Table 1, our model's cheat-corrected variance estimates are substantially better second-order calibrated than other methods, without sacrificing first-order calibration or predictive accuracy. In particular, most other methods tend to underestimate in-distribution epistemic uncertainty (with $\mathbb{E}[(\hat{p}^{\theta}_{Y|X} - p_{Y|X})^2] > \mathbb{E}[\hat{v}^{\theta}]$), although Naive NN overestimates it. We additionally train and evaluate models on a harder task variant, where we both add extra classes to make $p_{Y|X}$ more stochastic and also scramble the central image pixels to make underfitting more likely, and find that our method remains second-order calibrated, whereas other techniques become increasingly over- or under-confident. Of our two models, the SNGP variant performs the best suggesting that well-known techniques for improving first-order calibration also improve second-order calibration when training on pairs. We also point out that the NN Ensemble baseline gives similar variance estimates and similar ECE-1 values across the two task variants, but has worse ECE-2 and KL divergence scores on the harder variant. This means that ECE-1 and ensemble variance are not sufficient to identify tasks for which the model is a bad fit for $p_{Y|X}$, whereas the cheat-corrected variance estimate of our method is more representative of model quality. Further details for these experiments are provided in Appendix F.2.

6.2. English Descriptions of Digits of π

We next demonstrate that our technique can be directly applied to tasks with large output spaces such as sequence modeling. We construct a synthetic language modeling task that allow us to control the difficulty and amount of stochasticity in the target responses, where the goal is to correctly respond to requests like $x =$ "Tell me about digit 24 of π". Early digits of π are sampled more often than later ones, and the target responses are randomly-chosen true statements, such as "Sure, that is the number 6", "That's an even number", "It is spelled S I X", or "Sure, it's spelled with three letters", which are sampled with different probabilities and exhibit variation in both style and semantic content.

We train a 19M-parameter transformer model (Vaswani et al., 2017) from scratch for 50k iterations, tokenizing and concatenating the query X and two sampled responses Y_1 and Y_2 for each example. We next sample 120 statements from $\hat{p}^{\theta}_{Y_1|X}$ for each digit offset from 1 to 3,000, and label each sample as a statistical hallucination if $p_{Y|X}(y|x) = 0$ (e.g. if it is not a true statement about the requested digit). We then evaluate how well the bound in Theorem 4.5 holds

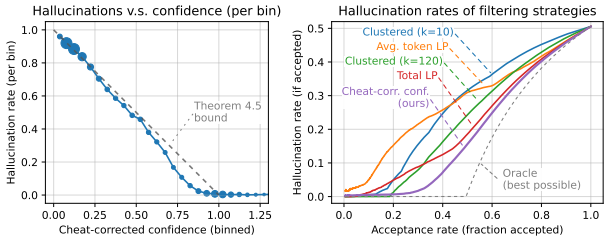

Figure 5. Left: For our digits-of-π model, binning samples by $C^{\theta}_{\text{CHEAT}}$ shows that hallucination rate is usually $\leq 1 - C^{\theta}_{\text{CHEAT}}$ as predicted by Theorem 4.5, although occasionally $C^{\theta}_{\text{CHEAT}} > 1$ due to miscalibration. *Right:* Ranking samples by $|1 - C^{\theta}_{\text{CHEAT}}(y|x)|$ yields a similar or lower hallucination rate than other common filtering strategies when applied to this model.

in practice by dividing samples into bins based on their predicted confidence $C^{\theta}_{\text{CHEAT}}$ and computing the fraction of samples in each bin that were hallucinated. Figure 5 (left) shows that the fraction of hallucinated samples is generally slightly lower than $1 - C^{\theta}_{\text{CHEAT}}$, as predicted by the bound in Theorem 4.5. However, somewhat surprisingly, we observe that $C^{\theta}_{\text{CHEAT}}(y|x) > 1$ for some samples, which would not occur for a well-calibrated model. Samples with $C^{\theta}_{\text{CHEAT}}$ slightly above 1 are usually correct, but in rare cases we also observe very large values of $C^{\theta}_{\text{CHEAT}}$ (e.g. about $\approx 10^4$ or larger), which tend to happen when the sampled Y_1 was malformed and out-of-distribution. We believe this stems from the inherent difficulty of making calibrated predictions over the space of all (pairs of) sequences. In practice, we suggest to use $|1 - C^{\theta}_{\text{CHEAT}}|$ for thresholding as an alternative to $1 - C^{\theta}_{\text{CHEAT}}$, which is equivalent if the model is calibrated. We explore other thresholding options and show specific examples where $C^{\theta}_{\text{CHEAT}}(y|x) > 1$ in Appendix F.3.

We next compare different strategies for distinguishing correct and hallucinated samples: ranking by the log-probability of each sample under the model (Total LP), ranking by length-normalized log-probability (Avg. Token LP) (Malinin & Gales, 2020), clustering semantically-equivalent answers in groups of k samples and thresholding by cluster size (Clustered) (Kuhn et al., 2022; Li et al., 2022a), and using our *cheat-based selective filtering* strategy from Section 4.1 (modified to threshold by $|1 - C^{\theta}_{\text{CHEAT}}(y|x)| \leq \beta$). We implement correctness and semantic equivalence checks using a lookup table, as described in Appendix F.3. Figure 5 (right) shows that filtering by our confidence measure allows generation of more responses with a lower hallucination rate relative to previously-proposed methods.

6.3. Safe Offline RL With Unobserved Confounders

Finally, we show as a proof of concept that our approach can detect confounders when doing imitation learning in POMDPs and thus avoid the "self-delusions" described by Ortega et al. (2021). We focus on the "Frozen Lake" grid-

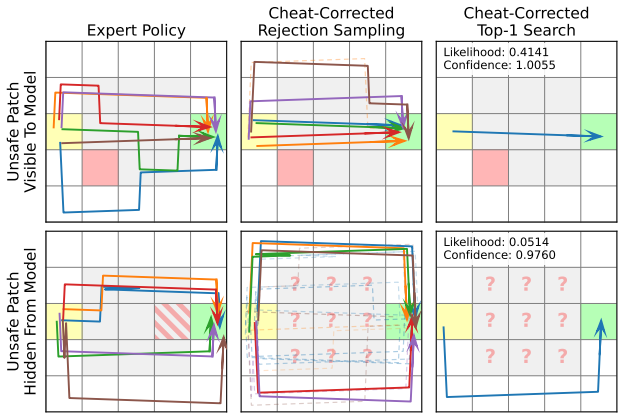

Figure 6. Our cheat-corrected decoding strategies (with $\beta = 0.05$) avoid unsafe actions in the "Frozen Lake" task. When the unsafe patch (red square) is visible, model samples imitate the expert distribution, and the highest-likelihood path crosses the lake. When it is hidden, $C^{\theta}_{\text{CHEAT}}$ is low for possibly-unsafe sampled paths (dashed lines), so our decoding strategies reject them in favor of safe paths.

world task (Warrington et al., 2020), where agents can take shortcuts across a lake to reach the goal, but a random part of the lake is unsafe to cross in each episode. We train a model to imitate expert demonstrations, where the experts always know and avoid the location of the unsafe patch, but the model can only see it 50% of the time. This partial-observation setting is an extreme example of misspecification, and can be viewed as a restriction on $\Phi(X)$: the model is forbidden from using part of the "true" input. Naive imitation learning in this setting would cause the model to learn to cross the lake randomly, which would be unsafe.

We train an 85M-parameter Transformer to imitate pairs of tokenized trajectories (Y_1, Y_2) drawn randomly from the expert policy, where the two demonstrations always share the same location of the unsafe patch. We then apply two of our cheat-corrected decoding strategies (rejection sampling and top-1 search) with the constraint $|1 - C^{\theta}_{\text{CHEAT}}| \leq 0.05$, and visualize the resulting trajectories in Figure 6. Our strategies behave like ordinary sampling and top-1 search when the unsafe location is visible to the model, but reject paths that cross the lake when the location is hidden, since any such path might have $p_{Y|X}(y|x) = 0$. Only the always-safe paths that avoid the lake are kept, since the model is confident that $p_{Y|X}(y|x) \approx \hat{p}^{\theta}_{Y_1|X}(y|x)$ for those trajectories.

7. Discussion

We have presented a principled new approach for identifying the gaps between a model $\hat{p}^{\theta}_{Y|X}$ and the ground truth $p_{Y|X}$, based on a remarkable equivalence between second-order calibration and pair prediction, and proven that calibrated pair predictors can be used to construct provably-correct bounds on $p_{Y|X}$. We have further demonstrated that our scheme is practically effective on both classification and sequence-modeling tasks, even without perfect calibration over $\mathcal{Y} \times \mathcal{Y}$. Although paired responses may not be available for all datasets, collecting paired fine-tuning data may still be easier than applying architecture-dependent uncertainty quantification strategies, especially for large models. We are optimistic that our procedure will scale up to this use case, and are eager to explore this direction in future work.

Acknowledgements

We would like to thank Dami Choi for helping with an early prototype of the idea, and Gustaf Ahdritz, Nikhil Vyas, Zelda Mariet, and Zi Wang for useful discussions. We are also thankful to Ayoub El Hanchi, David Glukhov, Stephan Rabanser, and Jasper Snoek for providing valuable feedback on the paper draft. Resources used in preparing this research were provided in part by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), RGPIN-2021-03445.

Impact Statement

Our work proposes a general strategy for training a model $\hat{p}^{\theta}_{Y|X}$ to accurately report how well it is able to fit an arbitrary process $p_{Y|X}$ on a per-input (or, precisely, per-equivalence-class) level. A large number of machine learning problems can be posed in this form, and the consequences of applying our technique would likely depend on the particular application. Overall, however, we hope our technique will make it easier to build safer and more reliable machine learning systems, by ensuring that they avoid taking unsafe actions or making unfair decisions when they are unable to accurately perform their intended tasks.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Angelopoulos, A. N. and Bates, S. A gentle introduction to conformal prediction and distribution-free uncertainty

quantification. *arXiv preprint arXiv:2107.07511*, 2021.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Barber, R. F. Is distribution-free inference possible for binary regression? *arXiv: Statistics Theory*, 2020.

Bengs, V., Hüllermeier, E., and Waegeman, W. Pitfalls of epistemic uncertainty quantification through loss minimisation. *Advances in Neural Information Processing Systems*, 35:29205–29216, 2022.

Bengs, V., Hüllermeier, E., and Waegeman, W. On second-order scoring rules for epistemic uncertainty quantification. *arXiv preprint arXiv:2301.12736*, 2023.

Berman, A. and Shaked-Monderer, N. *Completely positive matrices*. World Scientific, 2003.

Bernardo, J., Berger, J., Dawid, A., Smith, A., et al. Regression and classification using Gaussian process priors. *Bayesian statistics*, 6:475, 1998.

Bertsch, A., Xie, A., Neubig, G., and Gormley, M. R. It's mbr all the way down: Modern generation techniques through the lens of minimum bayes risk. *arXiv preprint arXiv:2310.01387*, 2023.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, 6, 1993.

Błasiok, J., Gopalan, P., Hu, L., and Nakkiran, P. When does optimizing a proper loss yield calibration? *ArXiv*, abs/2305.18764, 2023.

Cantelli, F. P. Sui confini della probabilita. In *Atti del Congresso Internazionale dei Matematici: Bologna del 3 al 10 de settembre di 1928*, pp. 47–60, 1929.

Carrell, A., Mallinar, N. R., Lucas, J., and Nakkiran, P. The calibration generalization gap. *ArXiv*, abs/2210.01964, 2022.

Chedzoy, O. B. Phi-Coefficient. In Kotz, S., Read, C. B., Balakrishnan, N., and Vidakovic, B. (eds.), *Encyclopedia of Statistical Sciences*. Wiley, 2 edition, December 2005. ISBN 9780471150442 9780471667193. doi: 10.1002/0471667196.ess1960.pub2. URL https://onlinelibrary.wiley.com/doi/10.1002/0471667196.ess1960.pub2.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Chen, X., Aksitov, R., Alon, U., Ren, J., Xiao, K., Yin, P., Prakash, S., Sutton, C., Wang, X., and Zhou, D. Universal self-consistency for large language model generation. *ArXiv*, abs/2311.17311, 2023.

Chollet, F. et al. Keras. https://keras.io, 2015.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *ArXiv*, abs/2110.14168, 2021.

Collier, M., Jenatton, R., Kokiopoulou, E., and Berent, J. Transfer and marginalize: Explaining away label noise with privileged information. In *International Conference on Machine Learning*, pp. 4219–4237. PMLR, 2022.

Dawid, A. P. The well-calibrated Bayesian. *Journal of the American Statistical Association*, 77(379):605–610, 1982.

Dawid, A. P. Present position and potential developments: Some personal views statistical theory the prequential approach. *Journal of the Royal Statistical Society: Series A (General)*, 147(2):278–290, 1984.

Dawid, A. P. Calibration-based empirical probability. *The Annals of Statistics*, 13(4):1251–1274, 1985.

DeepMind, Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovsky, P., Budden, D., Cai, T., Clark, A., Danihelka, I., Dedieu, A., Fantacci, C., Godwin, J., Jones, C., Hemsley, R., Hennigan, T., Hessel, M., Hou, S., Kapturowski, S., Keck, T., Kemaev, I., King, M., Kunesch, M., Martens, L., Merzic, H., Mikulik, V., Norman, T., Papamakarios, G., Quan, J., Ring, R., Ruiz, F., Sanchez, A., Sartran, L., Schneider, R., Sezener, E., Spencer, S., Srinivasan, S., Stanojević, M., Stokowiec, W., Wang, L., Zhou, G., and Viola, F. The DeepMind JAX Ecosystem, 2020. URL http://github.com/google-deepmind.

Ding, P. A first course in causal inference. *arXiv preprint arXiv:2305.18793*, 2023.

Durasov, N., Dorndorf, N., Le, H., and Fua, P. Zigzag: Universal sampling-free uncertainty estimation through

two-step inference. *arXiv preprint arXiv:2211.11435*, 2022.

Foster, D. P. and Vohra, R. V. Asymptotic calibration. *Biometrika*, 85(2):379–390, 1998.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, 2015.

Garrabrant, S., Benson-Tilsen, T., Critch, A., Soares, N., and Taylor, J. Logical induction. *arXiv preprint arXiv:1609.03543*, 2016.

Goan, E. and Fookes, C. Bayesian neural networks: An introduction and survey. *ArXiv*, abs/2006.12024, 2020.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Gupta, C. and Ramdas, A. Distribution-free calibration guarantees for histogram binning without sample splitting. In *International Conference on Machine Learning*, pp. 3942–3952. PMLR, 2021.

Gupta, C., Podkopaev, A., and Ramdas, A. Distribution-free binary classification: prediction sets, confidence intervals and calibration. *Advances in Neural Information Processing Systems*, 33:3711–3723, 2020.

Gutmann, M. and Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 297–304. JMLR Workshop and Conference Proceedings, 2010.

Hébert-Johnson, Ú., Kim, M. P., Reingold, O., and Rothblum, G. N. Multicalibration: Calibration for the (computationally-identifiable) masses. In *International Conference on Machine Learning*, 2018.

Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., and Lakshminarayanan, B. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019.

Hensman, J., Matthews, A., and Ghahramani, Z. Scalable variational Gaussian process classification. In *Artificial Intelligence and Statistics*, pp. 351–360. PMLR, 2015.

Hoeffding, W. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pp. 409–426, 1994.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Dai, W., Madotto, A., and Fung, P. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55:1 – 38, 2022.

Kadavath, S., Conerly, T., Askell, A., Henighan, T. J., Drain, D., Perez, E., Schiefer, N., Dodds, Z., DasSarma, N., Tran-Johnson, E., Johnston, S., El-Showk, S., Jones, A., Elhage, N., Hume, T., Chen, A., Bai, Y., Bowman, S., Fort, S., Ganguli, D., Hernandez, D., Jacobson, J., Kernion, J., Kravec, S., Lovitt, L., Ndousse, K., Olsson, C., Ringer, S., Amodei, D., Brown, T. B., Clark, J., Joseph, N., Mann, B., McCandlish, S., Olah, C., and Kaplan, J. Language models (mostly) know what they know. *ArXiv*, abs/2207.05221, 2022.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Kalai, A. T. and Vempala, S. S. Calibrated language models must hallucinate. *ArXiv*, abs/2311.14648, 2023.

Kendall, A. and Gal, Y. What uncertainties do we need in Bayesian deep learning for computer vision? *ArXiv*, abs/1703.04977, 2017.

Krizhevsky, A. Learning multiple layers of features from tiny images. In *Tech report*, 2009.

Kuhn, L., Gal, Y., and Farquhar, S. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*, 2022.

Kull, M. and Flach, P. Novel decompositions of proper scoring rules for classification: Score adjustment as precursor to calibration. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part I 15*, pp. 68–85. Springer, 2015.

Kumar, A., Liang, P. S., and Ma, T. Verified uncertainty calibration. *Advances in Neural Information Processing Systems*, 32, 2019.

Lahlou, S., Jain, M., Nekoei, H., Butoi, V. I., Bertin, P., Rector-Brooks, J., Korablyov, M., and Bengio, Y. DEUP: Direct epistemic uncertainty prediction. *arXiv preprint arXiv:2102.08501*, 2021.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Neural Information Processing Systems*, 2016.

Li, Y., Choi, D. H., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Tom, Eccles, Keeling, J., Gimeno, F.,

Lago, A. D., Hubert, T., Choy, P., de, C., d'Autume, M., Babuschkin, I., Chen, X., Huang, P.-S., Welbl, J., Gowal, S., Alexey, Cherepanov, Molloy, J., Mankowitz, D. J., Robson, E. S., Kohli, P., de, N., Freitas, Kavukcuoglu, K., and Vinyals, O. Competition-level code generation with AlphaCode. *Science*, 378:1092 – 1097, 2022a.

Li, Y., Lin, Z., Zhang, S., Fu, Q., Chen, B., Lou, J.-G., and Chen, W. Making language models better reasoners with step-aware verifier. In *Annual Meeting of the Association for Computational Linguistics*, 2022b.

Lin, Z., Khetan, A., Fanti, G., and Oh, S. Pacgan: The power of two samples in generative adversarial networks. *Advances in neural information processing systems*, 31, 2018.

Liu, J. Z., Lin, Z., Padhy, S., Tran, D., Bedrax-Weiss, T., and Lakshminarayanan, B. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *ArXiv*, abs/2006.10108, 2020.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Maddox, W. J., Garipov, T., Izmailov, P., Vetrov, D. P., and Wilson, A. G. A simple baseline for Bayesian uncertainty in deep learning. In *Neural Information Processing Systems*, 2019.

Malinin, A. and Gales, M. Predictive uncertainty estimation via prior networks. *Advances in neural information processing systems*, 31, 2018.

Malinin, A. and Gales, M. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*, 2020.

Muralidharan, O. and Najmi, A. Second order calibration: A simple way to get approximate posteriors. *arXiv preprint arXiv:1510.08437*, 2015.

Nado, Z., Band, N., Collier, M., Djolonga, J., Dusenberry, M. W., Farquhar, S., Feng, Q., Filos, A., Havasi, M., Jenatton, R., et al. Uncertainty baselines: Benchmarks for uncertainty & robustness in deep learning. *arXiv preprint arXiv:2106.04015*, 2021.

Narimatsu, H., Ozawa, M., and Kumano, S. Collision probability matching loss for disentangling epistemic uncertainty from aleatoric uncertainty. In *International Conference on Artificial Intelligence and Statistics*, pp. 11355–11370. PMLR, 2023.

Ni, A., Iyer, S., Radev, D. R., Stoyanov, V., tau Yih, W., Wang, S. I., and Lin, X. V. LEVER: learning to verify language-to-code generation with execution. *ArXiv*, abs/2302.08468, 2023.

Oakes, D. Self-calibrating priors do not exist. *Journal of the American Statistical Association*, 80:339–339, 1985.

OpenAI. GPT-4 technical report, 2023.

Ortega, P. A., Kunesch, M., Delétang, G., Genewein, T., Grau-Moya, J., Veness, J., Buchli, J., Degrave, J., Piot, B., Perolat, J., et al. Shaking the foundations: delusions in sequence models for interaction and control. *arXiv preprint arXiv:2110.10819*, 2021.

Osband, I., Wen, Z., Asghari, M., Ibrahimi, M., Lu, X., and Roy, B. V. Epistemic neural networks. *ArXiv*, abs/2107.08924, 2021.

Park, S., Li, S., Bastani, O., and Lee, I. PAC confidence predictions for deep neural network classifiers. *ArXiv*, abs/2011.00716, 2020.

Perez-Lebel, A., Morvan, M. L., and Varoquaux, G. Beyond calibration: estimating the grouping loss of modern neural networks. *arXiv preprint arXiv:2210.16315*, 2022.

Peterson, J. C., Battleday, R. M., Griffiths, T. L., and Russakovsky, O. Human uncertainty makes classification more robust. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9617–9626, 2019.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rasmussen, C. E. and Williams, C. K. I. Classification. In *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005. ISBN 9780262256834. doi: 10.7551/mitpress/3206.003.0006. URL https://doi.org/10.7551/mitpress/3206.003.0006.

Sale, Y., Bengs, V., Caprio, M., and Hüllermeier, E. Second-order uncertainty quantification: A distance-based approach. *arXiv preprint arXiv:2312.00995*, 2023.

Sandroni, A., Smorodinsky, R., and Vohra, R. V. Calibration with many checking rules. *Math. Oper. Res.*, 28:141–153, 2003.

Schmarje, L., Grossmann, V., Zelenka, C., Dippel, S., Kiko, R., Oszust, M., Pastell, M., Stracke, J., Valros, A., Volkmann, N., et al. Is one annotation enough?-a data-centric image classification benchmark for noisy and ambiguous label estimation. *Advances in Neural Information Processing Systems*, 35:33215–33232, 2022.

Schulman, J., Chen, X., and Abbeel, P. Equivalence between policy gradients and soft Q-learning. *arXiv preprint arXiv:1704.06440*, 2017.

Sensoy, M., Kandemir, M., and Kaplan, L. M. Evidential deep learning to quantify classification uncertainty. *ArXiv*, abs/1806.01768, 2018.

Shafer, G. and Vovk, V. *Game-theoretic foundations for probability and finance*, volume 455. John Wiley & Sons, 2019.

Vaicenavicius, J., Widmann, D., Andersson, C., Lindsten, F., Roll, J., and Schön, T. Evaluating model calibration in classification. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 3459–3467. PMLR, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vovk, V., Shen, J., Manokhin, V., and Xie, M.-g. Nonparametric predictive distributions based on conformal prediction. In *Conformal and probabilistic prediction and applications*, pp. 82–102. PMLR, 2017.

Wang, X., Wei, J., Schuurmans, D., Le, Q., hsin Chi, E. H., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171, 2022.

Warrington, A., Lavington, J. W., Scibior, A., Schmidt, M. W., and Wood, F. D. Robust asymmetric learning in POMDPs. In *International Conference on Machine Learning*, 2020.

Waudby-Smith, I. and Ramdas, A. Estimating means of bounded random variables by betting. *arXiv preprint arXiv:2010.09686*, 2020.

Wei, J., Zhu, Z., Cheng, H., Liu, T., Niu, G., and Liu, Y. Learning with noisy labels revisited: A study using real-world human annotations. *ArXiv*, abs/2110.12088, 2021.

Wen, Z., Osband, I., Qin, C., Lu, X., Ibrahimi, M., Dwaracherla, V., Asghari, M., and Van Roy, B. From predictions to decisions: The importance of joint predictive distributions. *arXiv preprint arXiv:2107.09224*, 2021.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533. PMLR, 2020.

Zadrozny, B. and Elkan, C. 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, 2002.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zarrieß, S., Voigt, H., and Schüz, S. Decoding methods in neural language generation: a survey. *Information*, 12(9): 355, 2021.

A. Sample Visualizations

In this section we present samples from our models for the Digits of Pi and Frozen Lake tasks (Sections 6.2 and 6.3).

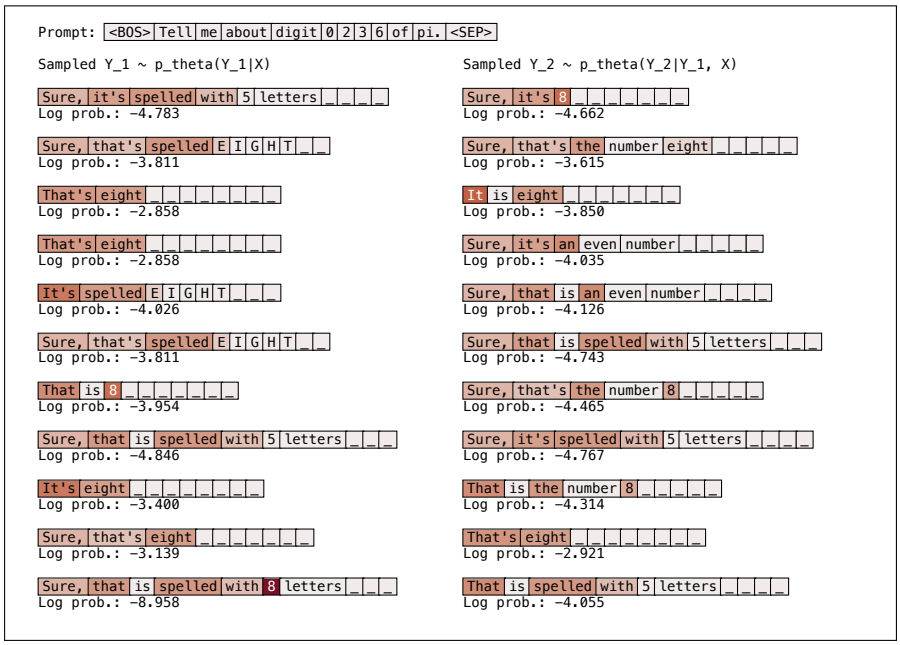

Figure 7. Results of sampling pairs (Y_1, Y_2) from the model $\hat{p}^{\theta}_{Y_1,Y_2|X}$ when asked about the 236th digit of π, which it "knows" is eight. (Our method does not actually require sampling Y_2; we show samples for illustrative purposes only.) Color denotes likelihood, with red denoting less-likely tokens. The left column is Y_1 and the right column is Y_2 (drawn conditional on Y_1); each row is an independent pair of samples for the prompt at the top. Note that the last row's Y_1 is a low-probability mistake which was sampled due to the high temperature. (The model "knows" eight is spelled with 5 letters, so Y_2 is inconsistent with Y_1).

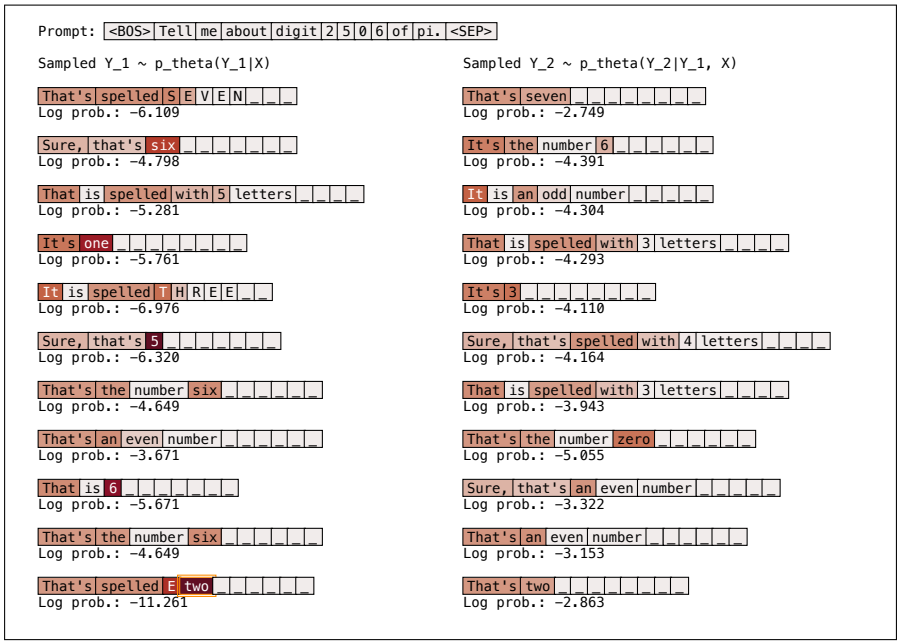

Figure 8. Results of sampling pairs (Y_1, Y_2) from the model $\hat{p}^{\theta}_{Y_1,Y_2|X}$, when asked about the 2506th digit of π (which it has not learned). The sampled Y_2 is usually consistent with the Y_1 sample, indicating that the model is "cheating" well. The last sample of Y_1 is malformed due to sampling a low-probability token.

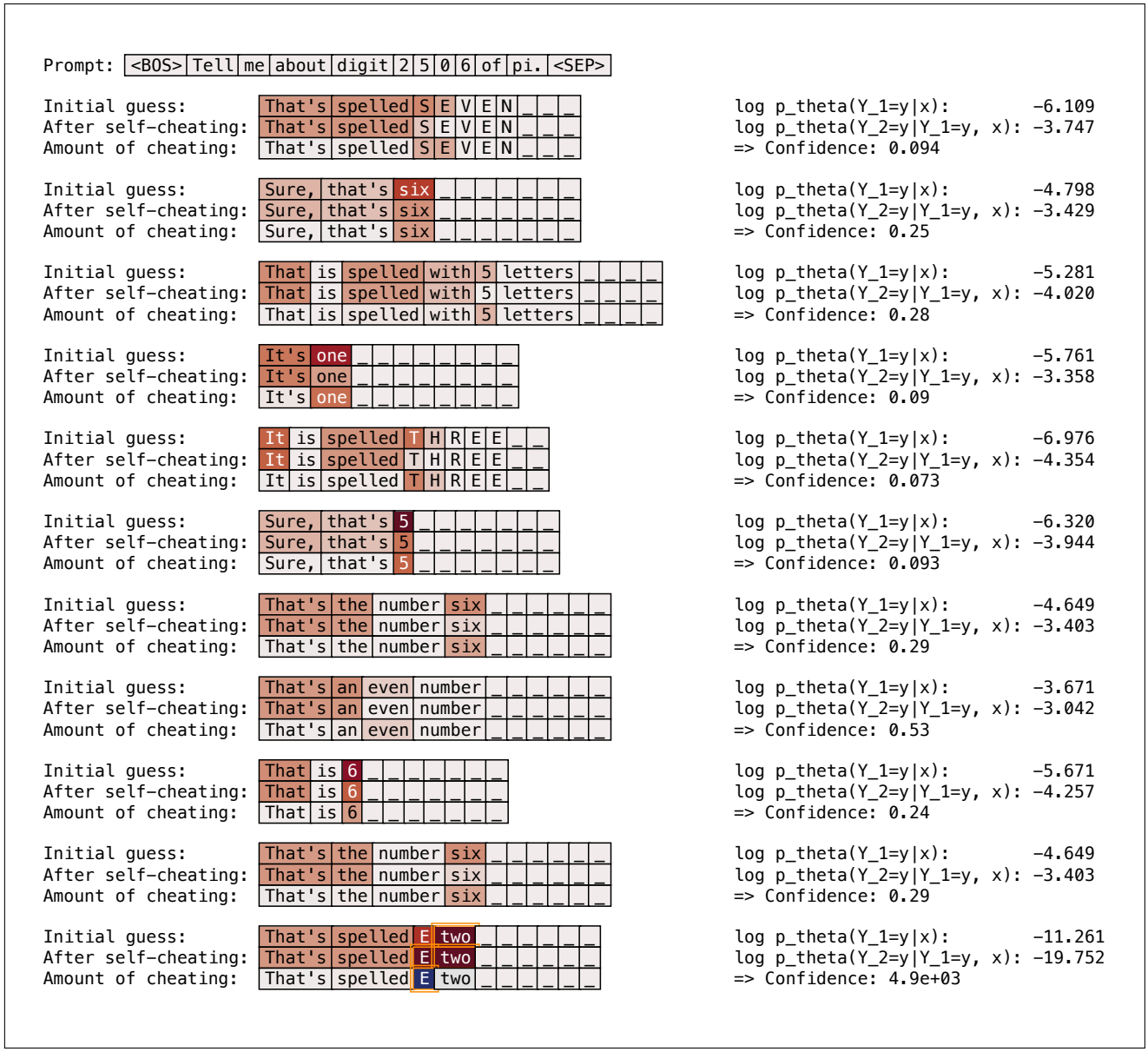

Figure 9. Scoring samples using our cheat-corrected epistemic confidence, for the 2506th digit of π (which it has not learned). Conditioning on $Y_1 = y$ reveals information about this digit, so the log probability increases when outputting $Y_2 = y$, and we can use the magnitude of the increase as a measurement of confidence. We can also attribute this increase to individual tokens (with red in the "Amount of cheating:" rows indicating a token whose likelihood increased after cheating). The last sample is malformed, so has very low probability as either Y_1 or Y_2, which leads to an outlier confidence greater than 1. We recommend discarding samples with confidences significantly larger than one, e.g. by keeping only those with $|1 - C^{\theta}_{\text{CHEAT}}| \leq \beta$.

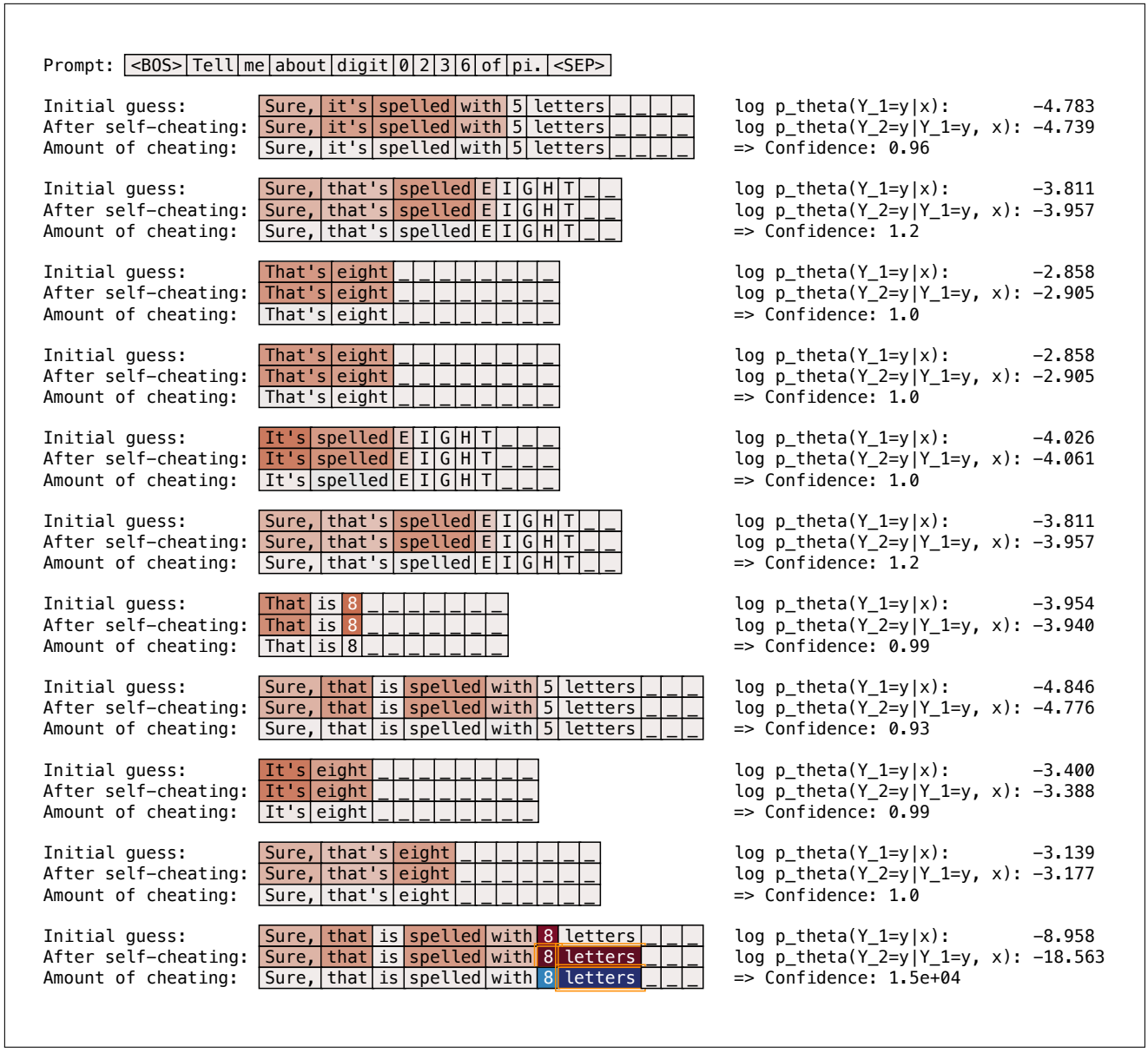

Figure 10. Scoring samples using our cheat-corrected epistemic confidence, for the 236th digit of π (which it "knows"). We repeat each response twice, comparing the probabilities $\hat{p}^{\theta}_{Y_1|X}(y|x)$ and $\hat{p}^{\theta}_{Y_2|Y_1,X}(y|y,x)$. Color indicates log probability for "Initial guess" and "After self-cheating", and differences between log probabilities for "Amount of cheating". For this prompt, conditioning on Y_1 does not significantly change the prediction of the model, because it already knows the value of the 236th digit. However, in the last sample, the probability decreases because the originally-sampled guess was a mistake (see Figure 7), leading to an outlier confidence value greater than one.

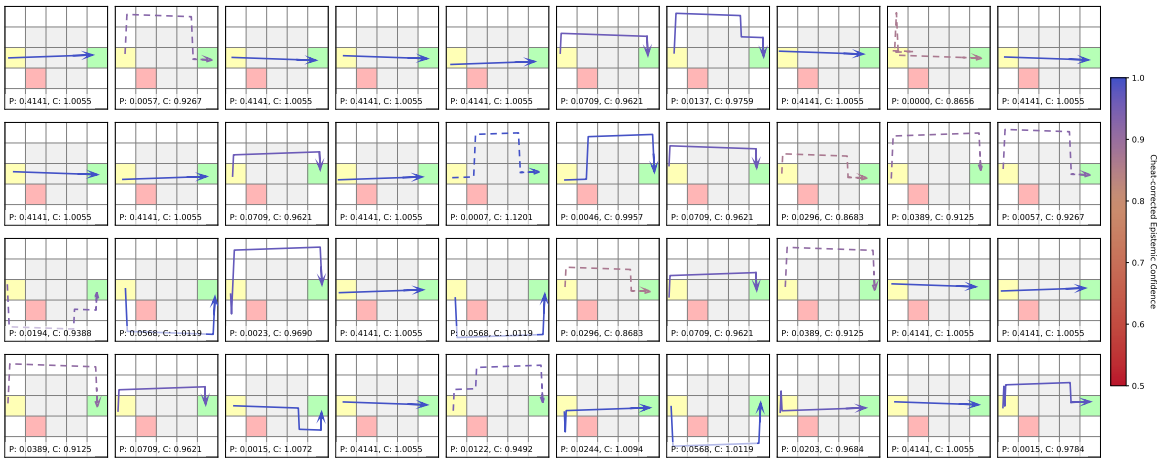

Figure 11. Model samples and confidences for the fully-observable version of the "Frozen Lake" task, with the unsafe patch in the bottom left. Dashed trajectories indicate samples that we would reject using a $|1 - C^{\theta}_{\text{CHEAT}}| \leq 0.05$ threshold. We add a small diagonal offset when plotting so that it is easier to follow paths that backtrack; the model itself only predicts discrete actions (left, right, up, down) and moves between grid cells. There is a fair amount of diversity among samples, although our strict decoding strategy does occasionally reject safe paths.

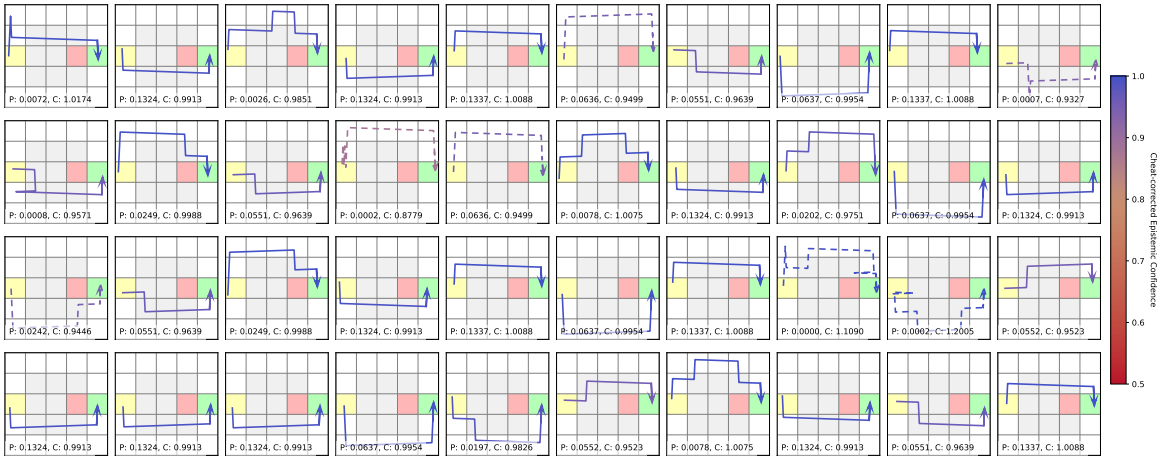

Figure 12. Model samples and confidences for the fully-observable version of "Frozen Lake" with the unsafe patch in the middle right.

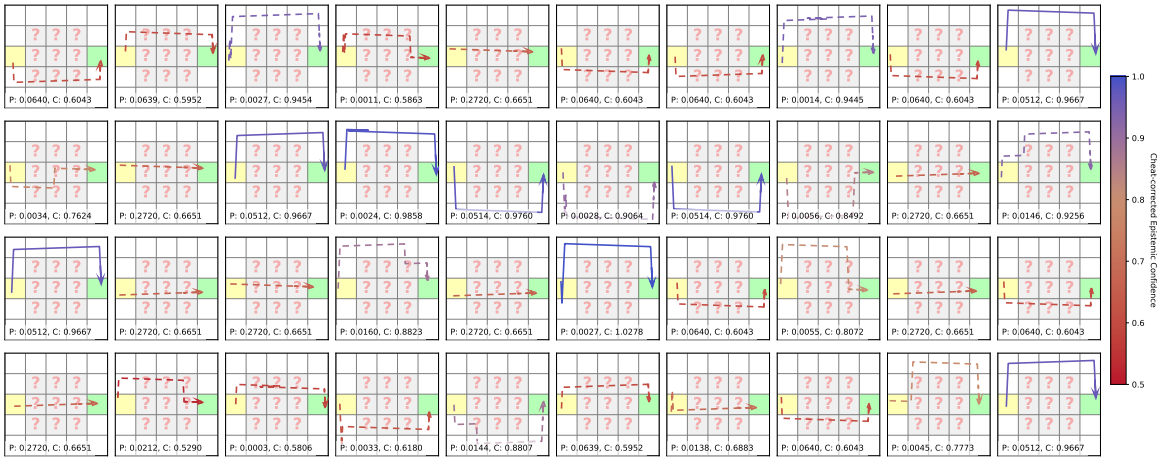

Figure 13. Model samples and confidences for "Frozen Lake" when the unsafe patch is hidden. Note that samples that cross the lake have much lower confidence when the unsafe patch is hidden, relative to similar trajectories in Figures 11 and 12.

B. Our Distribution-Free Confidence Intervals

In this section, we show the results of applying Theorem 4.6 to the 1D binary regression problem in Figure 3.

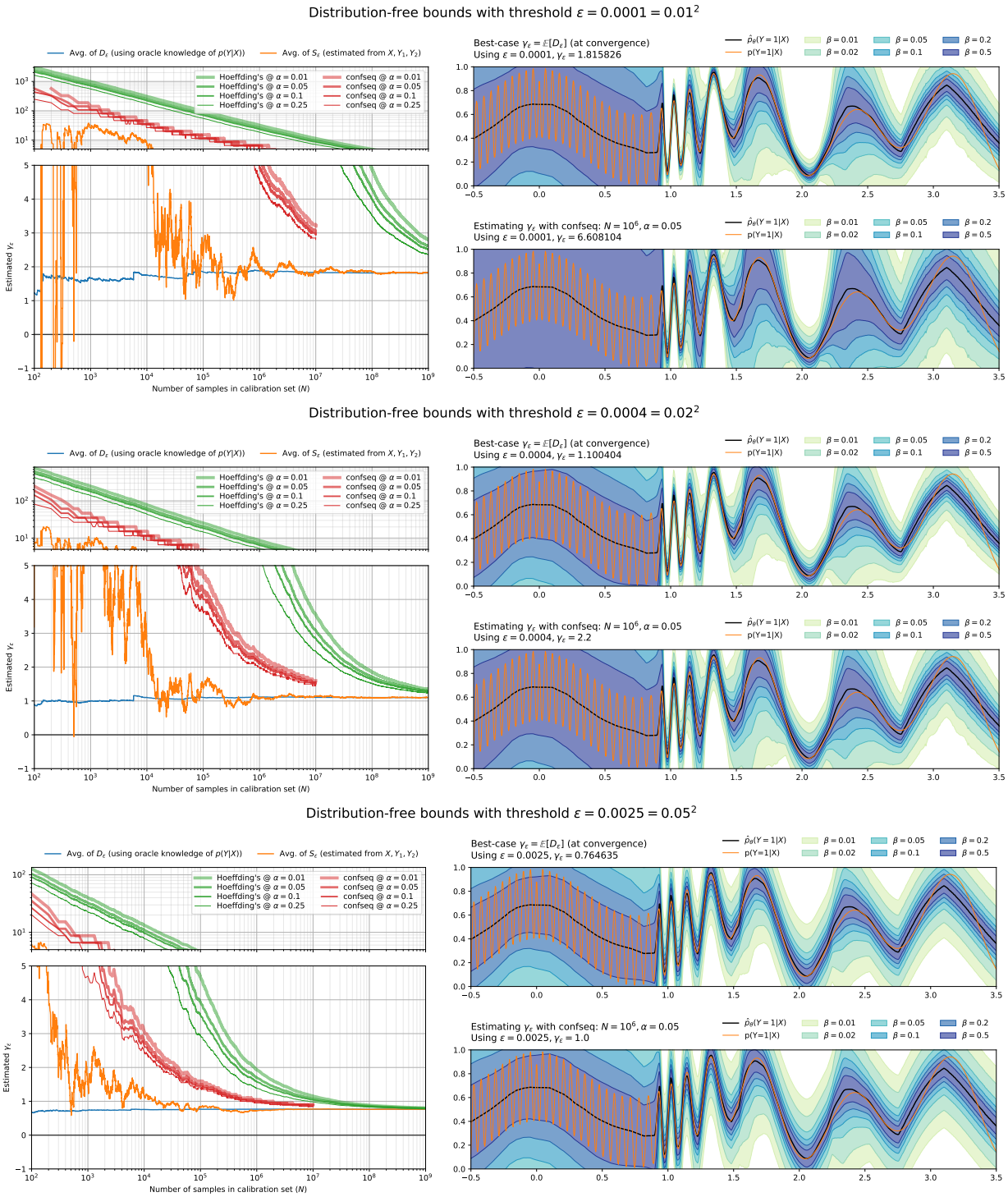

Figure 14. Visualization of our distribution-free bound for the toy 1-D binary regression problem in Figure 3, with ε set to $0.01^2, 0.02^2$, or 0.05^2. Left: Convergence of γ_ε based on Hoeffding's inequality and `confseq`, with running averages of D_ε and S_ε for reference. Right: Resulting confidence intervals for $p(Y|X)$, using either the best-case $\gamma_\varepsilon = \mathbb{E}[D_\varepsilon]$ or a value of γ_ε returned by Algorithm 1.

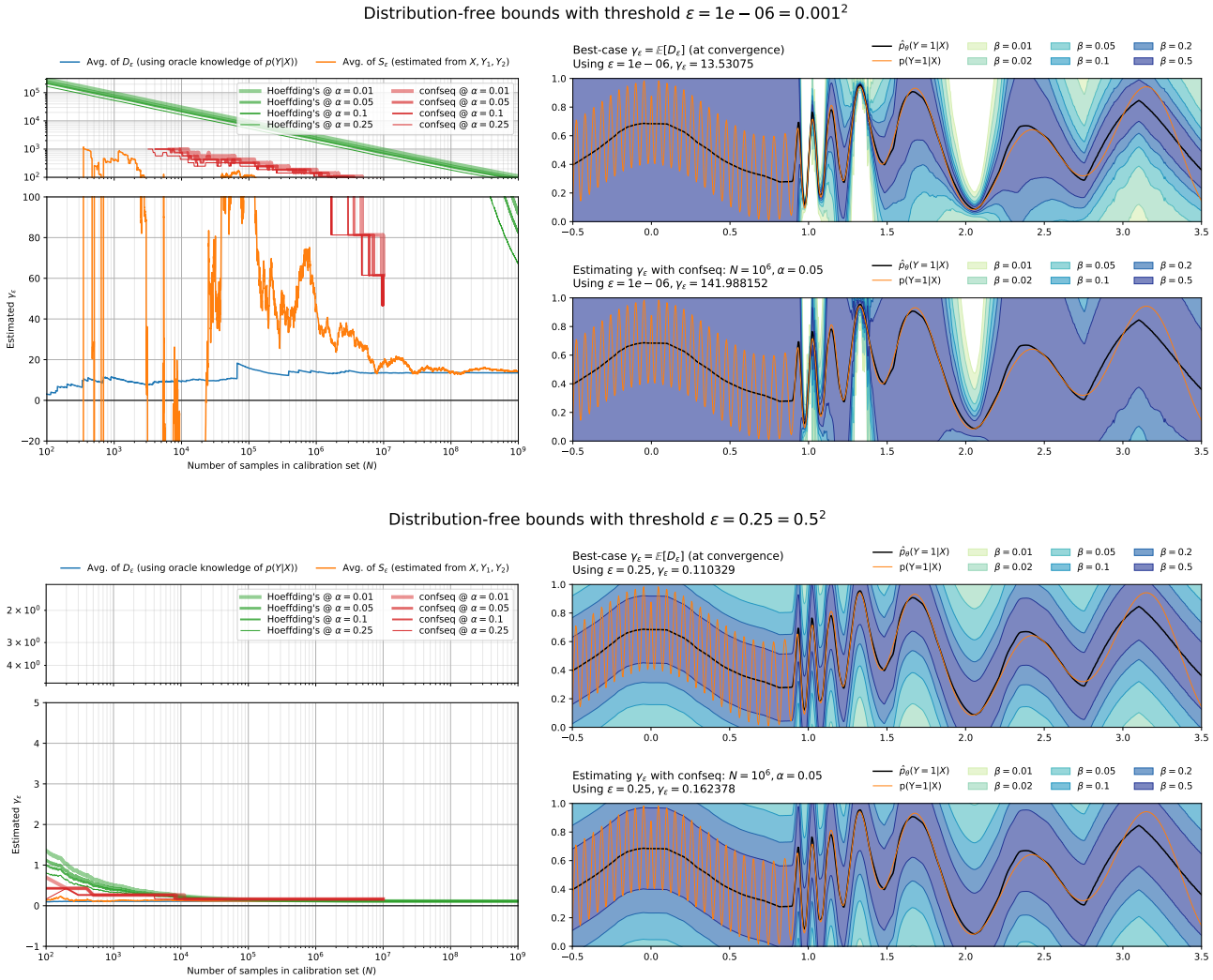

Figure 15. Visualization of our distribution-free bound with ε set to 0.001^2 (leading to a blowup of γ_ε and a very pessimistic bound) or 0.5^2 (for which the bound ignores v_θ^{CHEAT} entirely and has a constant width for all X, because $\hat{V}^\theta_{\text{CHEAT}}(y|x) \leq 0.5^2$ everywhere).

We hold $(\hat{p}^\theta_{Y_1|X}, \hat{\mathbf{\Sigma}}^\theta_{Y_1,Y_2|X})$ fixed (to the "Cheat-corrected NN" described in Appendix F.1), and study the behavior of the bound for different confidence interval algorithms, variance thresholds ε, failure tolerances α, and calibration set sizes N. We compare two confidence interval algorithms, Hoeffding's inequality (Hoeffding, 1994), discussed in Appendix D.4, and `confseq` (Waudby-Smith & Ramdas, 2020), desribed below.

In Figures 14 and 15, we enumerate $\varepsilon \in \{0.001, 0.01, 0.02, 0.05, 0.5\}$ and plot values of γ_ε as the number of calibration set examples N ranges from 100 to 10^9. For comparison, we also plot running average estimates of $\mathbb{E}[D_\varepsilon]$ (which require knowledge of $p(Y|X)$) and of $\mathbb{E}[S_\varepsilon]$ (as computed in Algorithm 1). We also plot the resulting confidence intervals for $p(Y|X)$ at various confidence levels β, using either an estimate based on 10^6 calibration set examples or the best-possible value $\mathbb{E}[D_\varepsilon]$ at convergence (computed using oracle knowledge of $p(Y|X)$).

Overall, we observe that `confseq`'s bounds are considerably tighter than those based on Hoeffding's inequality. Because our model $\hat{p}^\theta_{Y_1,Y_2|X}$ is not perfectly calibrated on pairs, setting ε too small leads to a blowup of $\mathbb{E}[D_\varepsilon]$ and an ineffective bound. Smaller ε also reduces the rate at which γ converges, so setting it to a larger value may be necessary if there is a limit on the size of the calibration set. On the other hand, setting ε too large produces confidence intervals that are the same width everywhere, ignoring $\hat{V}^\theta_{\text{CHEAT}}$ and instead using the marginal variance of $p_{Y|X}(1|X)$ across all X.

Note that, regardless of ε, the resulting bounds are provably correct with high probability (in the sense described by Theorem 4.6). However, when ε is set too small, it is more likely that the bound is overly conservative, and when ε is too

large, the coverage guarantees are more likely "trade off" errors between values of X, assigning conservative bounds to some regions and under-covered bounds to others so that the overall coverage target β is met. This can be seen in the plot for $\varepsilon = 0.25$. (In a sense *every* such bound must trade off errors between values of X, because after fixing $p_{Y|X}$, for each x the value of $p_{Y|X}(y|x)$ is either in the interval or not. But if ε is small and the model is well calibrated, this trading-off only occurs between examples in the same equivalence class, i.e. with the same value of $\Phi(x)$.)

confseq implementation details: For our `confseq` bounds, we use the `betting_cs` function from the `confseq` Python package,[2] which implements the algorithm described by Waudby-Smith & Ramdas (2020). We rescale our S_ε values so that they are bounded between 0 and 1, as assumed by the algorithm. `betting_cs` maintains a finite set of hypotheses about $\mathbb{E}[S_\varepsilon]$ and uses hypothesis testing to reject them; these hypotheses are evenly spaced over the unit interval by default, but we modify it slightly to choose a set of hypotheses that are more-closely concentrated around 0.5, which gives higher precision for $\gamma_\varepsilon \approx 0$ after inverting our rescaling. (The finite hypothesis set is the reason for the discrete jumps in the estimates produced by `confseq` in Figures 14 and 15.) We configure `betting_cs` with a prior mean of 1 and a prior variance of $\frac{0.5^4}{\varepsilon^2}$ for S_ε, which correspond to a prior mean of $\frac{1}{2} + \frac{1}{2\varepsilon}$ and prior variance of 0.5^6 after rescaling S_ε to the unit interval. Since betting-based confidence intervals are more computationally expensive than confidence intervals from Hoeffding's inequality, we only run `confseq` for calibration set sizes smaller than 10^7.

[2] https://github.com/gostevehoward/confseq

C. Additional discussion and related work

Pair prediction (two Y for the same X) v.s. "joint prediction" (different Xs and Ys) for epistemic uncertainty: As motivation for the Epinet uncertainty-quantification technique, Osband et al. (2021) have argued that uncertainty-aware agents should be judged not based on their uncertainty about Y for individual inputs X, but instead based on their joint distribution over a sequence of $Y^{(i)}$ drawn for a sequence of inputs $X^{(i)}$; they refer to this as "joint prediction". While somewhat similar to our proposed pair-prediction formalism in terms of motivation, the focus and applicability of the approaches is quite different.

Probabistic structure: The joint prediction criterion assumes the model has some hierarchical structure, such that it is possible to express a joint distribution over the outcomes $Y^{(i)}$ for different inputs $X^{(i)}$. Bayesian neural networks and Epinets satisfy this criterion, but not all neural networks express a joint in this way. In particular, a cheat-corrected pair-prediction neural network (as we propose) does not assume any joint distribution over outcomes for different X; its outputs can be converted into well-calibrated pointwise estimates of variance, but each prediction is made pointwise (e.g. for this particular x or the $x$es in a particular equivalence class).

Evaluation criterion: Osband et al. (2021) propose to use joint prediction primarily as an evaluation metric, based on theoretical results showing that joint predictions perform well for decision making (Wen et al., 2021). This can be interpreted as measuring how quickly a model can "learn" from new data to improve its predictions on future data points. In contrast, our work focuses on pair prediction as a way to train a model to be second-order calibrated, which then lets us estimate how accurately a model predicts its distance from $p(Y|X)$. Our evaluation metric is then the pointwise calibration of the second-order estimates.

Training objective: Our proposed training objective in directly trains a model to predict pairs, and our distribution-free adjustment procedure also directly uses paired data. This ensures that our approach can be statistically valid even if the model is misspecified or computationally limited, but requires the data collection process to be modified. On the other hand, Osband et al. (2021) do not train their models based on a joint prediction objective, but instead show that a per-sample log-likelihood objective leads to good joint predictions under the assumption that the data was generated by a distribution with a specific known form. This implies that the Epinet training objective is not necessarily second-order calibrated or robust to misspecification, and we find empirical evidence of this in Section 6.1.

Pair prediction v.s. minimum Bayes risk / repeated sampling techniques: Many postprocessing-based techniques improving model outputs using multiple samples, including clustering-based approaches, can be interpreted as instances of *minimum Bayes risk* (MBR) decoding (Bertsch et al., 2023). In MBR decoding, after obtaining a model $\hat{p}_\theta$ approximating some generative process, actions are selected not based on their likelihood under the model, but instead based on some error function $L(y, y')$ that compares possible outputs; an output y' is "good" if it achieves a low error in expectation across alternative outputs y sampled from the model $\hat{p}_\theta(Y|X = x)$. For instance, L might return 1 if two outputs are semantically equivalent.

Although our approach and MBR decoding both draw repeated samples from a conditional distribution of Y given X, they differ on *which* distribution is sampled. In MBR decoding, the training data usually consists of only one Y drawn from $p(Y|X)$ for each X, but multiple samples are drawn from $\hat{p}_\theta(Y|X)$ and compared at inference time. In contrast, in our pair-prediction technique, the training data must consist of two samples Y_1, Y_2 drawn from $p(Y|X)$ for each X, but at inference time we can sample and score single outputs from $\hat{p}_\theta(Y|X)$.

MBR also requires specification of a task-relevant error function $L(y, y')$, and does not distinguish between aleatoric and epistemic uncertainty in $\hat{p}_\theta(Y|X)$, but instead distinguishes between "risky"/"unusual" and "safe"/"common" samples using $L(y, y')$. In contrast, our technique is task-agnostic and explicitly distinguishes aleatoric and epistemic uncertainty. Note that, because it is task-agnostic, our approach may report uncertainty about hard-to-predict parts of Y even if they are not relevant to the downsteram task, whereas MBR decoding can explicitly ignore the irrelevant parts when computing L.

Comparison to other distribution-free statistical guarantees: Our adjustment procedure in Section 4.2 has some similarities to previous algorithms for distribution-free prediction.

Conformal prediction is a particularly common and powerful form of distribution-free inference; see Angelopoulos & Bates (2021) for an introduction. In general, conformal prediction lifts a predictor of *points* (i.e. samples $y_i \in \mathcal{Y}$) into a predictor of *prediction sets* (subsets of $\mathcal{Y}$) such that, for a new input drawn from the same distribution, the result lies in the predicted

set with high probability. The basic idea is to associate a "conformal score" to each outcome in $\mathcal{Y}$ that measures how badly the predictor was wrong (e.g. the prediction error), estimate an upper quantile of the conformal scores for the actual outcomes in the dataset, then construct a prediction set by removing any observation whose conformal score would be higher than this quantile). This idea is closely linked to that of hypothesis testing.

Directly applying conformal prediction to a binary classification problem produces prediction sets that are subsets of $\mathcal{Y} = \{0, 1\}$, but this is not ideal if there is uncertainty about Y, because then the best prediction set will often be $\{0, 1\}$ itself, which is trivial and uninformative; similar issues may also arise for larger $\mathcal{Y}$ in high-uncertainty settings. Related to our work, Barber (2020) investigated the feasibility of constructing confidence intervals for the probability $p(Y = 1|X)$ instead of the samples Y themselves. Working under the assumption that the data consists of (X, Y) pairs and that each X is seen at most once, Barber proved that any confidence intervals for $p(Y = 1|X)$ must necessarily also be a prediction set for Y itself. In other words, the confidence interval cannot be a tight bound on the true $p(Y = 1|X)$, since it must include at least one of the endpoints 0 or 1 with high probability, and may need to cover the whole unit interval for highly-stochastic Y. (Our approach avoids this limitation by assuming each input X is seen twice.)

Gupta et al. (2020) study the relationship between calibration, grouping functions, confidence intervals, and predictive sets for binary classification problems. They introduce the notion of confidence intervals and predictive sets with respect to a function f, where f plays the same role as our grouping function Φ, and study methods for bounding the true expectation $\mathbb{E}[Y|f(X)]$. They prove that, in general, parametric recalibration methods cannot be distribution-free calibrated, but if outputs of f are discretized to a finite set of bins first, then it is possible to construct distribution-free confidence intervals for the conditional probability $\mathbb{E}[Y|f(X)]$. A similar guarantee about calibration error was given by Kumar et al. (2019), who also proposed an efficient combined scaling-binning scheme, and a refined analysis that allows re-using samples was also given by Gupta & Ramdas (2021).

We note that confidence intervals for the expectation $\mathbb{E}[Y|f(X)]$ with respect to $f(X)$ are not the same as confidence intervals for the true conditional probability $p(Y = 1|X) = \mathbb{E}[Y|X]$. Constructing a confidence interval for $\mathbb{E}[Y|f(X)]$ allows you to guarantee that your model is nearly first-order calibrated; it gives an interval of values that is likely to contain the answer to the question "across all of the inputs for which my model's output is ϕ, how many will have $Y = 1$?" However, it does not tell you whether your model is doing a good job at separating examples with different true conditional probabilities. In contrast, our procedure directly produces a confidence interval for $p(Y = 1|X)$; it gives an interval of values in answer to the question "what is the chance that $Y = 1$ for this specific x?" such that the answer is likely[3] to be correct for most[4] randomly-chosen x.

For an estimator with finitely-many bins, and an infinite number of recalibration examples, the confidence intervals for $\mathbb{E}[Y|f(X)]$ will eventually converge on the exact value of $\mathbb{E}[Y|f(X)]$. However, our confidence intervals for $p(Y = 1|X)$ may *never* converge to the exact value of $p(Y = 1|X)$ if the model is unable to distinguish Xs with different label probabilities. This is unavoidable, since our model may not have capacity to express $p(Y = 1|X)$ without additional assumptions (whereas a lookup table always has enough capacity to estimate $\mathbb{E}[Y|f(X)]$ over finitely many bins). Nevertheless, if we happen to be lucky, and our model *is* actually able to predict the exact value for $p(Y = 1|X)$ (and is both calibrated and confident about this prediction, i.e. $\hat{V}^{\theta}_{\text{CHEAT}}(y|x) = 0$), then our confidence intervals will converge to that value. Our procedure is thus *adaptive* to the complexity of the specific dataset being used while remaining correct without additional assumptions, a desirable property for a distribution-free algorithm.

Previous uses of "second-order calibration" terminology: The term "second-order calibration" has been previously used by Muralidharan & Najmi (2015) to refer to a particular method for adjusting an arbitrary system to give approximate posteriors over a latent real-valued parameter, under strong distributional assumptions. Here "calibration" is used in the sense of "a calibration procedure" rather than as "the property of being calibrated", and the goal is to distinguish error in estimating the parameter from the intrinsic noise in the response-generating process, using an estimate of variance. The calibration procedure relies on binning examples x based on the output t of a learned model, assuming that the real-valued true parameter θ of interest follows a simple parameteric family, and then fitting the parameters of the family for each bin separately using maximum (marginal) likelihood; this is an extension of (re)calibration procedures that try to post-process a non-calibrated model to make it more (first-order) calibrated (e.g. Kumar et al. (2019)). Since true data for the parameter is not available, the focus of the work is primarily on ensuring that the simple parametric model fits the data well rather than

[3] With probability at least $1 - \alpha$.

[4] At least $1 - \beta$ of them.

on measuring the accuracy of the variance estimates.

We are not aware of any previous work that uses "second-order calibration" to refer to a formally-defined *property* of a predictive model rather than to a technique for postprocessing an existing model, nor any that considers it in the sense of predicting the squared error between a predicted discrete distribution and an unknown ground-truth discrete distribution (e.g. for a classifier or generative model) without distributional assumptions.

Joint, marginal, and class-wise calibration: Our definition of first-order calibration requires that all elements of the output joint distribution match their true expectation conditional on a *single* grouping function (or, equivalently, conditional on the full *vector* of model outputs). This is sometimes referred to as being "jointly calibrated". There are also weaker definitions of calibration. Following the terminology of Perez-Lebel et al. (2022), "classwise calibrated" models make individually-calibrated binary predictions about each possible class y (Zadrozny & Elkan, 2002), and "top-label calibrated" models first identify a most likely label and then make a calibrated binary predictions about the correctness of that guess (Guo et al., 2017).

Our technique fundamentally requires making predictions about a *pair* of outcomes (y_1, y_2). In particular, it is not enough to make separately-calibrated predictions $\hat{p}^{\theta}_{Y_1|X}(y_1|x)$ and $\hat{p}^{\theta}_{Y_2|Y_1,X}(y_2|y_1,x)$, since the procedure works by comparing how much more likely any given outcome y would be to occur a second time. In principle, however, we could still transform a multi-class classification problem into a set of binary classification problems, then apply our technique to the binary problems. In this setting, instead of predicting a full joint $\hat{p}^{\theta}_{Y_1,Y_2|X}(Y_1, Y_2|x)$, we could introduce binary outcome variables O^y_i such that $O^y_i = 1$ whenever $Y_i = y$, then make a set of individually-calibrated pair predictions $p(O^y_1, O^y_2|x)$, one for each y. This could then be used to construct second-order marginally-calibrated versions of classwise calibration or top-label calibration. Note that this approach would still allow you to estimate the epistemic variance for any particular class, but would not tell you a full covariance matrix.

How much does the choice of X matter? Our work has assumed the existence of a joint distribution of variables X, Y given by a conditional $p_{Y|X}(Y|X)$ and a distribution of queries $P(X)$. However, a first-order calibrated model is free to condition on an arbitrary *function* of X instead of X itself. This means that, in a standard machine learning setup, there may be some metaphysical ambiguity about what X "really" refers to.

A concrete example of this is our "Frozen Lake" experiments, where we randomly sample an environment X, then occasionally hide information about the unsafe patch to obtain X_{PARTIAL}, and finally train a model to map X_{PARTIAL} to a distribution over expert trajectories Y. If all we care about is first-order calibration, we could think of the procedure that transforms X into X_{PARTIAL} as either being part of the model's grouping function Φ or as being part of the ambient probability space. Similarly, we could either think of this as learning to approximate $p(Y|X)$ with a misspecified model, or as learning to approximate $p(Y|X_{\text{PARTIAL}})$ directly. These two are in a sense equivalent, since they produce the same samples of X_{PARTIAL} and would use the same cross entropy loss over Y, and a perhaps more standard choice would be to think of the hidden information as being some other variable Z, and think of the "true conditional" the model is learning as being $p(Y|X_{\text{PARTIAL}})$. You could make a similar argument for ordinary classifiers also, e.g. are feature normalization or augmentation strategies part of the data distribution or are they part of the model?

Once we involve second-order calibration, however, this distinction becomes practical rather than metaphysical: the query X is whatever information makes the two responses Y_1, Y_2 independent and identically distributed. In other words, if we construct a process that samples two responses Y_1, Y_2 that are i.i.d. given Z, the true conditional we are estimating will then be the conditional $p(Y|Z)$ regardless of what transformation we apply to Z before giving it to our model.

We believe this is a powerful strength of our approach, because it allows you to specify the "boundaries" of your desired conditional distribution by example rather than by assumption. If you wish to imitate a set of experts, you can collect a dataset by asking those experts, and any "common knowledge" that those experts have will become "part of X", regardless of whether or not you can encode it as part of the input to the model itself; a pair-predictor model will thus be incentivised to estimate whether or not it also knows that common knowledge. Similarly, anything that is independent between those experts will be treated as part of the aleatoric uncertainty in Y, since it cannot be used to help predict the answer of a different expert.

How tight is Theorem 4.5 (hallucination rate)? Our results in Theorem 4.5 provide an upper bound on the rate of statistical hallucinations when using a sufficiently well-behaved decoding algorithm. A natural question is whether this

bound is tight, and in what circumstances.

The bound in Theorem 4.5 will be tight if there are exactly two values for $p_{Y|X}$ conditioned on what the model "knows": zero, and some nonzero constant value. In this case, all of the variance in $p_{Y|X}$ conditioned on $\Phi(X)$ is caused by these two point masses, and the ratio of probabilities in $C^{\theta}_{\text{CHEAT}}$ will tell you the fraction of inputs X for which $p_{Y|X}$ takes the nonzero value. This might be the case if the model is very confident about the probability of the particular response y assuming it is correct, but does not know whether or not y is correct. Our experiments in the digits-of-pi task (discussed in Appendix F.3) approximately satisfy this property, since the only thing that changes between digits is the set of statements that are correct; the probability of any given statement is consistent across all queries for which it is correct.

In more realistic scenarios, there may be other aspects that influence the probability of a given response other than its correctness, e.g. the model may not know something about the typical style of answers to a particular type of question. This will lead to increased variance in $p_{Y|X}$ and a lower confidence. The epistemic confidence may still be useful in those settings as a normalized measurement of uncertainty in general, but it will likely produce a conservative overestimate of the chance of hallucination in particular.

We also note that the bound in Theorem 4.5 is particularly simple because it attempts to bound the rate of generating statements whose ground truth probability was exactly zero. However, this bound is a special case of Cantelli's inequality (Cantelli, 1929), a more general upper bound on a random variable given its mean and variance. We demonstrate how to use this to construct other one-sided bounds in Appendix D.3.

Partial observability and misspecification for decision making: The general problem of decision making under uncertainty is a well-studied problem, with much analysis under the formalism of partially-observable Markov decision processes (POMDPs) (Kaelbling et al., 1998). Agents acting in POMDPs must perform inference about their unknown state, based on a limited view of the environment.

Of particular relevance to our work is *asymmetric* imitation learning: the problem of learning to correctly imitate expert demonstrations when the experts may have access to additional information not known to the imitation agent. Naive imitation can cause an agent to take unsafe or undesirable actions, while "deluding itself" into expecting that every action it takes will be safe; Ortega et al. (2021) demonstrate this problem and identify it as an instance of *confounding* in a causal graph. This problem can be avoided if all training data is collected under the imitation-learning policy, where the expert actions are queried but only the imitation-learner's action are used. Relatedly, Warrington et al. (2020) describe a procedure for modifying the *expert* policy so that it can be safely imitated. Unfortunately, these procedures require the ability to dynamically query or adjust the expert policy, which is not always possible.

In the "Frozen Lake" experiment, we used the same hidden location when drawing the two expert decisions, so that making calibrated predictions about pairs of expert trajectories would requires us to quantify the influence of that extra information. As discussed above, this is essentially folding the partial observability of the "Frozen Lake" experiments into the grouping function $\Phi(X)$.

We think this is an interesting perspective which may be useful for thinking about the behavior of misspecified agents more broadly: training a calibrated predictor is roughly the same as having an optimal predictor that only sees some restricted view of its input, so perhaps techniques that work under partial observability could also be extended to work for arbitrary calibrated models.

Conditional independence requirements in Pair prediction v.s. randomized causal effect estimation: Our technique fundamentally assumes that Y_1 and Y_2 are independent and identically distributed according to $p(Y|X)$ for each X. A straightforward way to ensure this holds is to sample Y_1 and Y_2 from an explicit process for generating Y from X, e.g. by querying a random human annotator for each. Unfortunately, if direct access to an explicit response process is not available, our technique may not be directly applicable unless conditional independence is satisfied in some other way.

This use of an explicit label process in some ways resembles the use of treatment assignment in randomized controlled trials, where treatments are explicitly chosen by a randomized algorithm to ensure that treatments are conditionally independent of the outcomes (Ding, 2023). In the case of causal inference, this randomness allows estimating average treatment effects without making assumptions about the causal mechanism that induces those effects. In the case of our pair-prediction technique, the randomness of the label process allows us to distinguish aleatoric uncertainty from underfitting without making assumptions about the form of the distribution $p(Y|X)$.

We note also that a variety of techniques have been proposed for estimating causal effects *without* control of the treatment-assignment process, usually by making assumptions about the causal structure of the naturally-occuring data; studies that estimate causal effects in this way are referred to as "observational" studies (Ding, 2023). Such techniques can be effective if correctly designed, but can produce incorrect estimates if the causal model is misspecified (due to not accounting for all confounders). Similarly, methods such as Bayesian inference can produce good uncertainty estimates without using paired Y data if they are well specified, but can fail if misspecified.

Handling uncertainty using privileged information: Collier et al. (2022) propose a technique (TRAM) for improving robustness to label noise by training on *privileged information*. At training time, they allow the later layers of a network to condition on information such as annotator IDs, which can help explain away label noise. At inference time, this privileged information can then be marginalized out.

Similar to our method, TRAM involves collecting additional data from the response process $p(Y|X)$ at training time, but does not require this additional information when scoring new inputs. However, the additional information in TRAM can be seen as "explaining away" the *aleatoric uncertainty* in the process, allowing the model to focus on learning the link between X and Y; this aleatoric variation is then added back in through marginalization. In contrast, our technique conditions on a separate sample y_1 when predicting y_2, which can roughly be seen as "explaining away" the *epistemic uncertainty*. The remaining noise in $\hat{p}_\theta(y_2|y_1, x)$ is likely aleatoric, so we can correct for it by dividing it out.

Calibration, forecasting, and game-theoretic probability: In machine learning, calibration is usually formulated and evaluated with respect to an i.i.d. distribution of inputs X and outcomes Y. However, much of the initial work on calibration focused instead on *sequential forecasting* (Dawid, 1984), where the inputs X arrive sequentially and may not be identically distributed, and the goal is to produce a sequence of forecasts that are calibrated in the long run (e.g. asymtotically) and also achieve good performance according to a scoring rule. Although our contributions are focused on the i.i.d. setting, and paired responses seem difficult to extend to the sequential-forecasting setting, we briefly review some of the results on calibration for sequential forecasts for the interested reader.

Dawid (1982) proved that a coherent Bayesian reasoner must assign probability 1 to being eventually well-calibrated on any sequence of outcomes. This is roughly because a coherent Bayesian must be certain about their own prior (over the set of possible sequences); observed miscalibration can sometimes provide evidence about the sequence but can never convince the Bayesian to change their inference algorithm. Dawid (1985) expanded the notion of calibration to range over all computable subsequences (akin to the definition of multicalibration in the i.i.d. setting (Hébert-Johnson et al., 2018)), and showed that any computable forecasting strategies that achieve this stronger notion of calibration must eventually agree with each other.

Unfortunately, Oakes (1985) showed that no deterministic algorithm can be calibrated on every sequence: given any deterministic forecasting strategy, there exists an adversarial distribution of sequences for which it is miscalibrated. Interestingly, Foster & Vohra (1998) proved that a (non-Bayesian) forecaster can achieve asymptotic calibration on every sequence if they are allowed to add noise to their forecasts independently of the adversarially-selected outcomes in the sequence, but this comes at the cost of higher prediction error on each sequence due to the added noise. Sandroni et al. (2003) strengthened this result, showing that there exist (randomized) computable forecasting strategies that are asymptotically calibrated over all computable subsets of any sequence.

The above works show that calibration can be formalized in both probabilistic and game-theoretic terms. Game theory can also be used as a foundation for probability theory and hypothesis testing (Shafer & Vovk, 2019; Waudby-Smith & Ramdas, 2020), and approaches based on betting can even be used to define coherent "probabilities" over logical implications (Garrabrant et al., 2016). A promising property of this kind of formalization is that it can naturally account for computational constraints, by restricting the computational capabilities of the reasoner or adversary; this is much more difficult to do from a purely Bayesian perspective.

We note that, although it is difficult to define a coherent probability system over logical statements that converges to the truth, it is fairly easy to produce nearly-calibrated predictions about the truth values of a fixed distribution of logical statements, as long as you are OK with taking a non-Bayesian perspective and having a large grouping loss: you can simply output the fraction of all statements that are true, optionally after partitioning the space of statements into groups. Our experiments with predicting digits of π are closer to this simple procedure than they are to the algorithm of Garrabrant et al. (2016), and we conjecture that observed logical reasoning errors in language models can be thought of as more complex versions of this simple procedure as well.

D. Details about and proofs of theoretical results

In this section, we prove our theoretical results and discuss their implications.

D.1. First-Order Calibration

We first prove that our definition of calibration is equivalent to the more specific definition used in previous work (Kumar et al., 2019; Vaicenavicius et al., 2019; Perez-Lebel et al., 2022). This result was previously shown by Gupta et al. (2020).

Proposition 2.2. *If Eqn. (1) holds for some fixed Φ, then it must also hold for $\Phi^\theta_{Y|X}: \mathcal{X} \to \mathbb{R}^{\mathcal{Y}}$, where $\Phi^\theta_{Y|X}(x)_y \triangleq \hat{p}^\theta_{Y|X}(y|x)$.*

Proof. Fix Φ and suppose $\hat{p}^\theta_{Y|X}(Y{=}y|X{=}x) = \mathbb{E}\big[p(Y{=}y|X) \;\big|\; \Phi(X) = \Phi(x)\big]$. Then

$$\begin{aligned}\Phi^\theta_{Y|X}(x) &= \Big[\; \hat{p}^\theta_{Y|X}(Y{=}y_1|X{=}x),\; \ldots,\; \hat{p}^\theta_{Y|X}(Y{=}y_{|\mathcal{Y}|}|X{=}x),\; \Big] \\ &= \Big[\; \mathbb{E}\big[p(Y{=}y_1|X) \;\big|\; \Phi(X) = \Phi(x)\big],\; \ldots,\; \mathbb{E}\big[p(Y{=}y_{|\mathcal{Y}|}|X) \;\big|\; \Phi(X) = \Phi(x)\big] \;\Big]\end{aligned}$$

Let $h(\phi)$ be the vector

$$\begin{aligned}h(\phi) &= \Big[\; \mathbb{E}\big[p(Y{=}y|X) \;\big|\; \Phi(X) = \phi\big] \;\Big]_{y\in\mathcal{Y}} \\ &= \Big[\; \mathbb{E}\big[p(Y{=}y_1|X) \;\big|\; \Phi(X) = \phi\big],\; \ldots,\; \mathbb{E}\big[p(Y{=}y_{|\mathcal{Y}|}|X) \;\big|\; \Phi(X) = \phi\big] \;\Big]\end{aligned}$$

and observe then that $\Phi^\theta_{Y|X}(x) = h(\Phi(x))$. It follows that, for any $x \in \mathcal{X}$ and $y_i \in \mathcal{Y}$,

$$\begin{aligned}\mathbb{E}\big[p(Y{=}y_i|X) \;\big|\; \Phi^\theta_{Y|X}(X) = \Phi^\theta_{Y|X}(x)\big] &= \mathbb{E}\big[p(Y{=}y_i|X) \;\big|\; h(\Phi(X)) = h(\Phi(x))\big] \\ &= \mathbb{E}\Big[\mathbb{E}\big[p(Y{=}y_i|X) \;\big|\; \Phi(X)\big] \;\Big|\; h(\Phi(X)) = h(\Phi(x))\Big] \\ &= \mathbb{E}\Big[h(\Phi(X))_i \;\Big|\; h(\Phi(X)) = h(\Phi(x))\Big] \\ &= h(\Phi(x))_i \\ &= \mathbb{E}\big[p(Y{=}y_i|X) \;\big|\; \Phi(X) = \Phi(x)\big] \\ &= \hat{p}^\theta_{Y|X}(Y{=}y_i|X{=}x).\end{aligned}$$

In words, conditioning on the output of a calibrated model $\hat{p}^\theta_{Y|X}$ instead of on a more refined grouping $\Phi(X)$ only combines equivalence classes ϕ that have the same conditional expected value of $p(Y|X)$, so the overall expected value doesn't change in the larger equivalence classes. □

D.2. Equivalence of Pair Calibration and Second-Order Calibration

We now prove our main result Theorem 3.2, which we restate below:

Theorem 3.2. *If $\hat{p}^\theta_{Y_1,Y_2|X}$ is first-order calibrated at predicting pairs (Y_1, Y_2), then its marginal $\hat{p}^\theta_{Y_1|X}$ and pair covariance $\hat{\mathbf{\Sigma}}^\theta_{Y_1,Y_2|X}$ are second-order calibrated at predicting $p_{Y|X}$. Moreover, this is a bijection: for any second-order-calibrated $(\hat{p}^{\theta'}_{Y|X}, \hat{\mathbf{\Sigma}}^{\theta'})$, there is a unique first-order-calibrated $\hat{p}^\theta_{Y_1,Y_2|X}$ with $\hat{p}^{\theta'}_{Y|X} = \hat{p}^\theta_{Y_1|X}$ and $\hat{\mathbf{\Sigma}}^{\theta'} = \hat{\mathbf{\Sigma}}^\theta_{Y_1,Y_2|X}$.*

Proof. Consider the mapping f defined by $f(\hat{p}^\theta_{Y_1,Y_2|X}) = (\hat{p}^\theta_{Y_1|X}, \hat{\mathbf{\Sigma}}^\theta_{Y_1,Y_2|X})$. We will show that f is a bijection between the set of first-order-calibrated $\hat{p}^\theta_{Y_1,Y_2|X}$ and the set of second-order-calibrated $(\hat{p}^{\theta'}_{Y|X}, \hat{\mathbf{\Sigma}}^{\theta'})$.

Recall that we can decompose the (conditional) covariance into a difference of expectations:

$$\begin{aligned}\mathrm{Cov}\Big[p_{Y|X}(y|X), p_{Y|X}(y'|X)\Big] &= \mathbb{E}\Big[\big(p_{Y|X}(y|X) - \mathbb{E}[p_{Y|X}(y|X)]\big)\big(p_{Y|X}(y'|X) - \mathbb{E}[p_{Y|X}(y'|X)]\big)\Big] \\ &= \mathbb{E}\Big[p_{Y|X}(y|X)p_{Y|X}(y'|X)\Big] - \mathbb{E}\Big[p_{Y|X}(y|X)\Big]\mathbb{E}\Big[p_{Y|X}(y'|X)\Big] \\ &= \mathbb{E}\Big[P(Y_1 = y|X)P(Y_2 = y'|X)\Big] - \mathbb{E}\Big[P(Y_1 = y|X)\Big]\mathbb{E}\Big[P(Y_2 = y'|X)\Big] \\ &= \mathbb{E}\Big[P(Y_1 = y, Y_2 = y'|X)\Big] - \mathbb{E}\Big[P(Y_1 = y|X)\Big]\mathbb{E}\Big[P(Y_2 = y'|X)\Big].\end{aligned}$$

This also holds when conditioned on a specific equivalence class $X \in [x]_\Phi$.

First suppose $\hat{p}^\theta_{Y_1,Y_2|X}(y_1, y_2|x)$ is a first-order-calibrated predictor of pairs with grouping function Φ, i.e.

$$\hat{p}^\theta_{Y_1,Y_2|X}(y_1, y_2|x) = \mathbb{E}\big[P(Y_1 = y_1, Y_2 = y_2|X) \;\big|\; X \in [x]_\Phi\big].$$

Marginalizing out y_2 gives

$$\begin{aligned}
\hat{p}^\theta_{Y_1|X}(y_1|x) &= \sum_{y_2} \mathbb{E}\big[P(Y_1 = y_1, Y_2 = y_2|X) \;\big|\; X \in [x]_\Phi\big] \\
&= \mathbb{E}\big[\sum_{y_2} P(Y_1 = y_1, Y_2 = y_2|X) \;\big|\; X \in [x]_\Phi\big] \\
&= \mathbb{E}\big[P(Y_1 = y_1|X) \;\big|\; X \in [x]_\Phi\big] \\
&= \mathbb{E}\big[p_{Y|X}(y_1|X) \;\big|\; X \in [x]_\Phi\big]
\end{aligned}$$

so $\hat{p}^\theta_{Y_1|X}$ is first-order calibrated at predicting Y. The same is true for $\hat{p}^\theta_{Y_2|X}$. We then also have

$$\begin{aligned}
\hat{\mathbf{\Sigma}}^\theta_{Y_1,Y_2|X}(x)_{y,y'} &= \hat{p}^\theta_{Y_1,Y_2|X}(y, y'|x) - \hat{p}^\theta_{Y_1|X}(y|x)\,\hat{p}^\theta_{Y_2|X}(y'|x) \\
&= \mathbb{E}\Big[P(Y_1 = y, Y_2 = y'|X) \;\Big|\; X \in [x]_\Phi\Big] \\
&\quad - \mathbb{E}\Big[P(Y_1 = y|X) \;\Big|\; X \in [x]_\Phi\Big]\mathbb{E}\Big[P(Y_2 = y'|X) \;\Big|\; X \in [x]_\Phi\Big] \\
&= \mathrm{Cov}\Big[p_{Y|X}(y|X), p_{Y|X}(y'|X) \;\Big|\; X \in [x]_\Phi\Big].
\end{aligned}$$

Thus $(\hat{p}^\theta_{Y_1|X}, \hat{\mathbf{\Sigma}}^\theta_{Y_1,Y_2|X})$ is second-order calibrated. We conclude that $f(\hat{p}^\theta_{Y_1,Y_2|X})$ is second-order calibrated whenever $\hat{p}^\theta_{Y_1,Y_2|X}$ is first-order calibrated (with the same grouping function Φ).

Now consider the mapping g that maps each second-order calibrated $(\hat{p}^{\theta'}_{Y|X}, \hat{\mathbf{\Sigma}}^{\theta'})$ to the $\hat{p}^\theta_{Y_1,Y_2|X}$ given by

$$\hat{p}^\theta_{Y_1,Y_2|X}(y_1, y_2|x) = \hat{\mathbf{\Sigma}}^{\theta'}(x)_{y_1,y_2} + \hat{p}^{\theta'}_{Y|X}(y_1|x)\,\hat{p}^{\theta'}_{Y|X}(y_2|x).$$

If $(\hat{p}^{\theta'}_{Y|X}, \hat{\mathbf{\Sigma}}^{\theta'})$ are second-order calibrated with respect to grouping function Φ, then we can expand this as

$$\begin{aligned}
\hat{p}^\theta_{Y_1,Y_2|X}(y_1, y_2|x) &= \mathrm{Cov}\Big[p_{Y|X}(y_1|X), p_{Y|X}(y_2|X) \;\Big|\; X \in [x]_\Phi\Big] + \mathbb{E}\big[p_{Y|X}(y_1|X) \;\big|\; X \in [x]_\Phi\big]\mathbb{E}\big[p_{Y|X}(y_2|X) \;\big|\; X \in [x]_\Phi\big] \\
&= \mathbb{E}\Big[P(Y_1 = y_1, Y_2 = y_2|X) \;\Big|\; X \in [x]_\Phi\Big].
\end{aligned}$$

This implies that $\hat{p}^\theta_{Y_1,Y_2|X}$ is a first-order-calibrated predictor of pairs, and thus that $g(\hat{p}^{\theta'}_{Y|X}, \hat{\mathbf{\Sigma}}^{\theta'})$ is first-order calibrated whenever $(\hat{p}^{\theta'}_{Y|X}, \hat{\mathbf{\Sigma}}^{\theta'})$ are second-order calibrated (with the same grouping function Φ).

Finally, to show that g is the inverse of f, let $\hat{p}^\theta_{Y_1,Y_2|X}$ be an arbitrary first-order calibrated predictor with grouping function Φ, and observe that

$$\begin{aligned}
\Big[g(f(\hat{p}^\theta_{Y_1,Y_2|X}))\Big](y, y'|x) &= \hat{\mathbf{\Sigma}}^\theta_{Y_1,Y_2|X}(x)_{y,y'} + \hat{p}^\theta_{Y_1|X}(y|x)\hat{p}^\theta_{Y_1|X}(y'|x) \\
&= \mathrm{Cov}\Big[p_{Y|X}(y|X), p_{Y|X}(y'|X) \;\Big|\; X \in [x]_\Phi\Big] \\
&\qquad + \mathbb{E}\big[p_{Y|X}(y|X) \;\big|\; X \in [x]_\Phi\big]\mathbb{E}\big[p_{Y|X}(y'|X) \;\big|\; X \in [x]_\Phi\big] \\
&= \mathbb{E}\Big[P(Y_1 = y, Y_2 = y'|X) \;\Big|\; X \in [x]_\Phi\Big] \\
&= \hat{p}^\theta_{Y_1,Y_2|X}(y, y'|x).
\end{aligned}$$

Thus $g(f(\hat{p}^\theta_{Y_1,Y_2|X})) = \hat{p}^\theta_{Y_1,Y_2|X}$, so f is a bijection with inverse $f^{-1} = g$. □

Since predictors can always reduce their expected loss (under a proper scoring rule) by becoming better calibrated on their training task, Theorem 3.2 implies that our pair-prediction procedure incentivizes second-order calibration.

D.3. Proofs of Error Bounds for Calibrated Pair Predictors

Theorem 4.2. *Suppose $\hat{p}^{\theta}_{Y_1,Y_2|X}$ is calibrated. Let A be any event and $\tilde{Y} \in \mathcal{Y}$ be any (possibly random) value such that $\tilde{Y}, A \perp\!\!\!\perp X \mid \Phi^{\theta}_{Y_1,Y_2|X}(X)$. Then*

$$\mathbb{E}\Big[\big(\hat{p}^{\theta}_{Y_1|X}(\tilde{Y}|X) - p_{Y|X}(\tilde{Y}|X)\big)^2 \,\Big|\, A\Big] = \mathbb{E}\Big[\hat{V}^{\theta}_{\text{CHEAT}}(\tilde{Y}|X) \,\Big|\, A\Big].$$

Furthermore, for any $\beta \in (0,1)$,

$$P\left[\left|\hat{p}^{\theta}_{Y_1|X}(\tilde{Y}|X) - p_{Y|X}(\tilde{Y}|X)\right| \geq \sqrt{\frac{\hat{V}^{\theta}_{\text{CHEAT}}(\tilde{Y}|X)}{\beta}} \;\middle|\; A\right] \leq \beta.$$

Proof. We will show that these properties hold individually for every value of $\Phi^{\theta}_{Y_1,Y_2|X}(X)$ and $\tilde{Y}$ conditioned on A, and so they must also hold overall.

Let ϕ and y be arbitrary, and x be such that $\Phi(x) = \phi$. Since $\hat{p}^{\theta}_{Y_1,Y_2|X}$ is calibrated, it must be symmetric, so $\hat{V}^{\theta}_{\text{CHEAT}}(y|X) = \hat{\mathbf{\Sigma}}^{\theta}_{Y_1,Y_2|X}(X)_{y,y}$. Furthermore $(\hat{p}^{\theta}_{Y_1|X}, \hat{\mathbf{\Sigma}}^{\theta}_{Y_1,Y_2|X})$ must be second-order calibrated, and $A, \tilde{Y}$ are independent of X given $\Phi(X) = \phi$, so

$$\hat{p}^{\theta}_{Y_1|X}(y|x) = \mathbb{E}\Big[p_{Y|X}(y|X) \;\Big|\; \Phi^{\theta}_{Y_1,Y_2|X}(X) = \phi\Big] = \mathbb{E}\Big[p_{Y|X}(y|X) \;\Big|\; \Phi^{\theta}_{Y_1,Y_2|X}(X) = \phi, \tilde{Y} = y, A\Big]$$
$$\hat{V}^{\theta}_{\text{CHEAT}}(y|x) = \mathrm{Var}\Big[p_{Y|X}(y|X) \;\Big|\; \Phi^{\theta}_{Y_1,Y_2|X}(X) = \phi\Big] = \mathrm{Var}\Big[p_{Y|X}(y|X) \;\Big|\; \Phi^{\theta}_{Y_1,Y_2|X}(X) = \phi, \tilde{Y} = y, A\Big]$$

Let B be the event where $(\Phi^{\theta}_{Y_1,Y_2|X}(X) = \phi, \tilde{Y} = y, A)$ all occur.

For the first part, we have

$$\begin{aligned}
&\mathbb{E}\Big[\big(\hat{p}^{\theta}_{Y_1|X}(y|X) - p_{Y|X}(y|X)\big)^2 \,\Big|\, B\Big] \\
&\quad= \mathbb{E}\left[\Big(\mathbb{E}\Big[p_{Y|X}(y|X) \;\Big|\; \Phi^{\theta}_{Y_1,Y_2|X}(X) = \phi, \tilde{Y} = y, A\Big] - p_{Y|X}(y|X)\Big)^2 \,\middle|\, B\right] \\
&\quad= \mathrm{Var}\Big[p_{Y|X}(y|X) \;\Big|\; \Phi^{\theta}_{Y_1,Y_2|X}(X) = \phi, \tilde{Y} = y, A\Big] \\
&\quad= \hat{V}^{\theta}_{\text{CHEAT}}(y|x) = \mathbb{E}\big[\hat{V}^{\theta}_{\text{CHEAT}}(y|X)\big|B\big]
\end{aligned}$$

where the last step follows because x is an arbitrary input with $\Phi^{\theta}_{Y_1,Y_2|X}(x) = \phi$ and every such x has the same value for $\hat{V}^{\theta}_{\text{CHEAT}}(y|x)$. Taking expectations over all values of X and $\tilde{Y}$ given A yields the desired result.

For the second part, Chebyshev's inequality ensures that

$$P\left[|\mathbb{E}[Z] - Z| \geq \sqrt{\frac{\mathrm{Var}(Z)}{\beta}}\right] \leq \beta$$

for any random variable Z and any β. Applying this with $Z = p_{Y|X}(y|X)$ conditioned on B gives

$$P\left[\left|\mathbb{E}[p_{Y|X}(\tilde{Y}|X)|B] - p_{Y|X}(\tilde{Y}|X)\right| \geq \sqrt{\frac{\mathrm{Var}\Big[p_{Y|X}(y|X) \;\Big|\; B\Big]}{\beta}} \;\middle|\; B\right] \leq \beta.$$

so

$$P\left[\left|\hat{p}^{\theta}_{Y_1|X}(y|x) - p_{Y|X}(\tilde{Y}|X)\right| \geq \sqrt{\frac{\hat{V}^{\theta}_{\text{CHEAT}}(y|x)}{\beta}} \;\middle|\; B\right] \leq \beta.$$

We can now marginalize over $\Phi^{\theta}_{Y_1,Y_2|X}(X)$ and $\tilde{Y}$ conditioned on A to obtain the desired result. □

Proposition 4.4. *If $\hat{p}^{\theta}_{Y_1,Y_2|X}$ is calibrated, then for any $x \in \mathcal{X}, y \in \mathcal{Y}$ we have $0 \leq C^{\theta}_{\text{CHEAT}}(y|x) \leq 1$, with $C^{\theta}_{\text{CHEAT}}(y|x) = 1$ if and only if $\hat{p}^{\theta}_{Y_1|X}(y|x) = p_{Y|X}(y|x)$.*

Proof. $C^{\theta}_{\text{CHEAT}}(y|x) \geq 0$ because both its numerator and denominator are nonnegative. Furthermore, if $C^{\theta}_{\text{CHEAT}}(y|x) = 1$, the numerator and denominator must be equal.

To show that $C^{\theta}_{\text{CHEAT}}(y|x) \leq 1$, algebraic manipulation allows us to write it in the form

$$C^{\theta}_{\text{CHEAT}}(y|x) = \frac{\hat{p}^{\theta}_{Y_1|X}(y|x)}{\hat{p}^{\theta}_{Y_2|Y_1,X}(y|y,x)} = \frac{\hat{p}^{\theta}_{Y_1|X}(y|x)^2}{\hat{p}^{\theta}_{Y_1,Y_2|X}(y,y|x)} = \left(\frac{\hat{p}^{\theta}_{Y_1,Y_2|X}(y,y|x)}{\hat{p}^{\theta}_{Y_1|X}(y|x)^2}\right)^{-1}$$
$$= \left(1 + \frac{\hat{p}^{\theta}_{Y_1,Y_2|X}(y,y|x) - \hat{p}^{\theta}_{Y_1|X}(y|x)^2}{\hat{p}^{\theta}_{Y_1|X}(y|x)^2}\right)^{-1} = \left(1 + \frac{\hat{V}^{\theta}_{\text{CHEAT}}(y|x)}{\hat{p}^{\theta}_{Y_1|X}(y|x)^2}\right)^{-1}.$$

If $\hat{p}^{\theta}_{Y_1,Y_2|X}$ is calibrated, it must be symmetric, so $\hat{V}^{\theta}_{\text{CHEAT}}(y|X) = \hat{\mathbf{\Sigma}}^{\theta}_{Y_1,Y_2|X}(X)_{y,y}$, which is a conditional variance and thus cannot be negative. It follows that $\frac{\hat{V}^{\theta}_{\text{CHEAT}}(y|x)}{\hat{p}^{\theta}_{Y_1|X}(y|x)^2} \geq 0$, so $C^{\theta}_{\text{CHEAT}}(y|x) \leq 1$. □

Note of caution: When $\hat{p}^{\theta}_{Y_1,Y_2|X}$ is *not* calibrated, it is no longer true that $\hat{V}^{\theta}_{\text{CHEAT}}(y|X)$ is necessarily equal to $\hat{\mathbf{\Sigma}}^{\theta}_{Y_1,Y_2|X}(X)_{y,y}$, because $\hat{p}^{\theta}_{Y_1,Y_2|X}$ is not necessarily symmetric. It is also not necessarily true that $\hat{\mathbf{\Sigma}}^{\theta}_{Y_1,Y_2|X}(X)_{y,y}$ is an epistemic variance, since Theorem 3.2 does not hold. This can mean that, for miscalibrated $\hat{p}^{\theta}_{Y_1,Y_2|X}$, it is possible to observe $\hat{V}^{\theta}_{\text{CHEAT}}(y|x) < 0$ and $C^{\theta}_{\text{CHEAT}} > 1$, and we do observe this in some of our experiments. We discuss this further in Appendix F.

Theorem 4.5. *Suppose $\hat{p}^{\theta}_{Y_1,Y_2|X}$ is calibrated. Let A be the event that a decoding algorithm responds to a query X, and $\tilde{Y} \in \mathcal{Y}$ be its response. If $A, \tilde{Y} \perp\!\!\!\perp X \mid \Phi^{\theta}_{Y_1,Y_2|X}(X)$, then the statistical hallucination rate of the generated responses is bounded above as*

$$P\left[p_{Y|X}(\tilde{Y}|X) = 0 \;\middle|\; A\right] \leq 1 - \mathbb{E}\left[C^{\theta}_{\text{CHEAT}}(\tilde{Y}|X) \;\middle|\; A\right].$$

Proof. Similar to our proof of Theorem 4.2, we can prove that this holds individually for every value of $\Phi^{\theta}_{Y_1,Y_2|X}(X)$ and $\tilde{Y}$ conditioned on A and then take an expectation. As before, let ϕ and y be arbitrary, and x be such that $\Phi(x) = \phi$. By Cantelli's inequality (Cantelli, 1929) (also known as the one-sided Chebyshev's inequality),

$$P\left[Z \leq \mathbb{E}[Z] - \lambda\right] \leq \frac{\text{Var}(Z)}{\text{Var}(Z) + \lambda^2}$$

for any random variable Z and any β. Substituting $\lambda = \mathbb{E}[Z]$,

$$P\left[Z \leq 0\right] \leq \frac{\text{Var}(Z)}{\text{Var}(Z) + \mathbb{E}[Z]^2} = \frac{\mathbb{E}[Z^2] - \mathbb{E}[Z]^2}{\mathbb{E}[Z^2]} = 1 - \frac{\mathbb{E}[Z]^2}{\mathbb{E}[Z^2]}$$

Now letting $Z = p_{Y|X}(y|X)$ and conditioning on $B = (\Phi^{\theta}_{Y_1,Y_2|X}(X) = \phi, \tilde{Y} = y, A)$ as before, and using the fact that $\hat{p}^{\theta}_{Y_1,Y_2|X}$ is calibrated,

$$P\left[p_{Y|X}(y|X) \leq 0 \mid B\right] \leq 1 - \frac{\mathbb{E}[p_{Y|X}(y|X)|B]^2}{\mathbb{E}[p_{Y|X}(y|X)^2|B]} = 1 - \frac{\mathbb{E}[p_{Y|X}(y|X)|\Phi(X) = \phi]^2}{\mathbb{E}[p_{Y|X}(y|X)^2|\Phi(X) = \phi]} = 1 - \frac{\hat{p}^{\theta}_{Y_1|X}(y|x)^2}{\hat{p}^{\theta}_{Y_1,Y_2|X}(y,y|x)}$$
$$= 1 - \frac{\hat{p}^{\theta}_{Y_1|X}(y|x)}{\hat{p}^{\theta}_{Y_2|Y_1,X}(y|y,x)} = 1 - C^{\theta}_{\text{CHEAT}}(y|x) = 1 - \mathbb{E}[C^{\theta}_{\text{CHEAT}}(y|X)|B]$$

where here x is an arbitrary input with $\Phi^{\theta}_{Y_1,Y_2|X}(x) = \phi$, since every such x has the same value for $C^{\theta}_{\text{CHEAT}}(y|x)$. Taking expectations of both sides over all values for ϕ and y completes the proof. □

As an aside, we note that it's possible to prove a one-sided bound on $p_{Y|X}(y|X)$ by combining the proof ideas from Theorem 4.2 and Theorem 4.5:

Proposition D.1. *Suppose $\hat{p}^{\theta}_{Y_1,Y_2|X}$ is calibrated. Let A be any event and $\tilde{Y} \in \mathcal{Y}$ be any (possibly random) output such that $\tilde{Y}, A \perp\!\!\!\perp X \mid \Phi^{\theta}_{Y_1,Y_2|X}(X)$. Then for any $\beta \in (0,1)$,*

$$P\left[p_{Y|X}(y|X) \leq \hat{p}^{\theta}_{Y_1|X}(y|X) - \sqrt{\hat{V}^{\theta}_{\text{CHEAT}}(y|x)\left(\frac{1}{\beta}-1\right)} \;\middle|\; A\right] \leq \beta.$$

Proof. As above, but let $\lambda = \sqrt{\hat{V}^{\theta}_{\text{CHEAT}}(y|x)\left(\frac{1}{\beta}-1\right)}$ in Cantelli's inequality. Then substituting $\hat{V}^{\theta}_{\text{CHEAT}}$ as before we obtain

$$P\big[p_{Y|X}(y|X) \leq \hat{p}^{\theta}_{Y_1|X}(y|X) - \lambda \;\big|\; B\big] \leq \frac{\hat{V}^{\theta}_{\text{CHEAT}}(y|x)}{\hat{V}^{\theta}_{\text{CHEAT}}(y|x) + \hat{V}^{\theta}_{\text{CHEAT}}(y|x)\left(\frac{1}{\beta}-1\right)} = \beta,$$

and taking expectations completes the proof. □

This is a better bound than the one in Theorem 4.2 in the case where we want a conservative estimate of how small $p_{Y|X}(y|X)$ could be with confidence $1-\beta$, instead of wanting to bound the distance to $\hat{p}^{\theta}_{Y_1|X}(y|X)$.

D.4. Distribution-Free Bounds on $p(Y|X)$

Suppose $\mathcal{Y} = \{0,1\}$. Our distribution-free high-probability bound on $p(Y|X)$ is based on the random variable

$$D_\varepsilon = \frac{\left(p_{Y|X}(1|X) - \hat{p}^{\theta}_{Y|X}(1|X)\right)^2}{\max\{\hat{V}^{\theta}(1|X), \varepsilon\}}.$$

D_ε is always nonnegative, and if $\hat{p}^{\theta}_{Y|X}$ and $\hat{V}^{\theta}$ are second-order well-calibrated (or, more precisely, if $\hat{V}^{\theta}$ is the diagonal of the epistemic covariance matrix) the expected value $\mathbb{E}[D_\varepsilon]$ should be at most 1.

The following lemma shows that we can use $\mathbb{E}[D_\varepsilon]$ to bound $p(Y|X)$, even if $\hat{p}^{\theta}_{Y|X}$ is not well calibrated:

Lemma D.2. *Suppose $\mathbb{E}[D_\varepsilon] \leq \gamma_\varepsilon$. Then for a randomly sampled input $X \sim p(X)$ and any $\beta \in [0,1)$, the true conditional $p(Y=1|X)$ lies within*

$$\hat{p}^{\theta}_{Y|X}(1|X) \pm \sqrt{\max\{\hat{V}^{\theta}(Y=1|X), \varepsilon\}\,\gamma_\varepsilon/\beta} \tag{2}$$

with probability at least $1-\beta$.

Proof. By Markov's inequality, for any $\beta \in [0,1)$, $p\Big(D_\varepsilon \geq \mathbb{E}[D_\varepsilon]/\beta\Big) \leq \beta$. In other words, with probability at least $1-\beta$, $D_\varepsilon < \mathbb{E}[D_\varepsilon]/\beta$. Since $\mathbb{E}[D_\varepsilon] \leq \gamma_\varepsilon$, this means that

$$\frac{\left(p_{Y|X}(1|X) - \hat{p}^{\theta}_{Y|X}(1|X)\right)^2}{\max\{\hat{V}^{\theta}(1|X), \varepsilon\}} < \gamma_\varepsilon/\beta$$

with probability at least $1-\beta$, in which case

$$\left|p_{Y|X}(1|X) - \hat{p}^{\theta}_{Y|X}(1|X)\right| < \sqrt{\max\{\hat{V}^{\theta}(1|X), \varepsilon\}\gamma_\varepsilon/\beta}.$$ □

We can use this to prove Theorem 4.6, which we restate below:

Theorem 4.6. *Let $\hat{p}^{\theta}_{Y|X}$, $\hat{V}^{\theta}$, and $p_{Y|X}$ be arbitrary. With probability at least $1-\alpha$ (over draws of the calibration set), Algorithm 1 returns a value γ^{+}_{ε} such that, for a randomly sampled input $X \sim p(X)$, and any $\beta \in (0,1), y \in \{0,1\}$,*

$$P\left[\left|\hat{p}^{\theta}_{Y|X}(y|X) - p_{Y|X}(y|X)\right| \geq \sqrt{\frac{\gamma^{+}_{\varepsilon}\max\{\hat{V}^{\theta}(y|X),\varepsilon\}}{\beta}}\right] \leq \beta.$$

Proof. It remains to show that the γ_ε produced by Algorithm 1 is an upper bound on $\mathbb{E}[D_\varepsilon]$, at which point we can apply Lemma D.2.

Define the random variable

$$S_\varepsilon = \frac{\big(Y_1 - \hat{p}^\theta_{Y|X}(1|X)\big)\big(Y_2 - \hat{p}^\theta_{Y|X}(1|X)\big)}{\max\{\hat{V}^\theta(1|X), \varepsilon\}}$$

and observe that

$$\begin{aligned}
\mathbb{E}[S_\varepsilon] &= \mathbb{E}\left[\frac{\big(Y_1 - \hat{p}^\theta_{Y|X}(1|X)\big)\big(Y_2 - \hat{p}^\theta_{Y|X}(1|X)\big)}{\max\{\hat{V}^\theta(1|X), \varepsilon\}}\right] \\
&= \mathbb{E}\left[\frac{\mathbb{E}\big[\big(Y_1 - \hat{p}^\theta_{Y|X}(1|X)\big)\big(Y_2 - \hat{p}^\theta_{Y|X}(1|X)\big) \;\big|\; X\big]}{\max\{\hat{V}^\theta(1|X), \varepsilon\}}\right] \\
&= \mathbb{E}\left[\frac{\big(\mathbb{E}[Y_1|X] - \hat{p}^\theta_{Y|X}(1|X)\big)\big(\mathbb{E}[Y_2|X] - \hat{p}^\theta_{Y|X}(1|X)\big)}{\max\{\hat{V}^\theta(1|X), \varepsilon\}}\right] \\
&= \mathbb{E}\left[\frac{\big(p(Y=1|X) - \hat{p}^\theta_{Y|X}(1|X)\big)^2}{\max\{\hat{V}^\theta(1|X), \varepsilon\}}\right] = \mathbb{E}[D_\varepsilon].
\end{aligned}$$

This means that any confidence interval for $\mathbb{E}[S_\varepsilon]$ is also a confidence interval for $\mathbb{E}[D_\varepsilon]$. Furthermore, we know that $-\frac{1}{\varepsilon} \le S_\varepsilon \le \frac{1}{\varepsilon}$, and we can construct samples of S_ε by using $\hat{p}^\theta_{Y|X}$ and samples (X, Y_1, Y_2), as described in Algorithm 1.

By assumption, the subroutine MEANCONFITVL constructs a confidence interval for the mean of a bounded random variable. In other words, it satisfies the property that, for any bounded i.i.d. random variables $V^{(i)} \in (a, b)$, if we let $(L, U) = \textsc{MeanConfItvl}\big(\{V^{(1)}, \dots, V^{(N)}\}, a, b, \alpha\big)$, then $L \le \mathbb{E}[V] \le U$ with probability at least α.

Algorithm 1 then applies MEANCONFITVL to the samples of $S^{(i)}_\varepsilon$, which are each bounded between $-1/\varepsilon$ and $1/\varepsilon$. As such, we know that the returned value γ^+_ε will satisfy $\mathbb{E}[S_\varepsilon] \le \gamma^+_\varepsilon$ with probability at least $(1-\alpha)$. This confidence interval must also be a bound on $\mathbb{E}[D_\varepsilon]$, so we can apply Lemma D.2, which completes the proof. □

There are multiple possible implementations of the subroutine MEANCONFITVL, based on different confidence intervals for the mean of a bounded random variable. A particularly simple implementation is based on Hoeffding's inequality (Hoeffding, 1994):

Proposition D.3 (Confidence interval via Hoeffding's inequality)**.** *In Algorithm 1, an implementation of subroutine* $\textsc{MeanConfItvl}\big(\{s^{(i)}_\varepsilon\}_{i=1}^N, -\frac{1}{\varepsilon}, \frac{1}{\varepsilon}, \alpha\big)$ *that returns the values*

$$\gamma^-_\varepsilon = -\frac{1}{\varepsilon}, \qquad \gamma^+_\varepsilon = \frac{1}{N}\sum_{i=1}^N s^{(i)}_\varepsilon + \sqrt{2\frac{-\log\alpha}{n\varepsilon^2}}$$

guarantees that $\gamma^-_\varepsilon \le \mathbb{E}[S_\varepsilon] \le \gamma^+_\varepsilon$ *with probability at least* $1-\alpha$ *(and thus that Theorem 4.6 holds).*

Proof. For the lower bound, we know that $\gamma^-_\varepsilon = -\frac{1}{\varepsilon} \le \mathbb{E}[S_\varepsilon]$ due to the boundedness of S_ε. For the upper bound, Hoeffding's inequality gives us

$$p\left(\mathbb{E}[S_\varepsilon] - \frac{1}{N}\sum_{i=1}^N s^{(i)}_\varepsilon \ge t\right) \le \exp\left(-\frac{nt^2\varepsilon^2}{2}\right).$$

Choosing $t = \sqrt{2\frac{-\log\alpha}{n\varepsilon^2}}$, this becomes

$$p\left(\mathbb{E}[S_\varepsilon] - \frac{1}{N}\sum_{i=1}^N s^{(i)}_\varepsilon \ge t\right) \le \alpha,$$

so we must have

$$\mathbb{E}[S_\varepsilon] \leq \frac{1}{N}\sum_{i=1}^{N} s_\varepsilon^{(i)} + t = \frac{1}{N}\sum_{i=1}^{N} s_\varepsilon^{(i)} + \sqrt{2\frac{-\log\alpha}{n\varepsilon^2}} = \gamma_\varepsilon^+$$

with probability at least $1-\alpha$. □

There are also more complex algorithms which may require fewer samples to give an accurate upper bound. For instance, Waudby-Smith & Ramdas (2020) describe an algorithm for constructing tighter confidence intervals based on "betting strategies". This algorithm is available in the `confseq` Python package[5]. See Appendix B for additional experiments studying the convergence of our bound in practice, and comparing the bounds constructed using Hoeffding's inequality to bounds using `confseq`.

D.4.1. Is Theorem 4.6 the best we can do without distributional assumptions?

Theorem 4.6 does not require that the model is perfectly calibrated. If $\hat{p}^\theta_{Y|X}$ and $\hat{V}^\theta$ are actually epistemically perfectly calibrated, and we take $\varepsilon \to 0$, we will have $\mathbb{E}[S_\varepsilon] = \mathbb{E}[D_\varepsilon] \to 1$, so in principle we can make Equation (2) arbitrarily close to Theorem 4.2 by choosing a small enough ε and a large enough calibration set. (This assumes that the confidence interval for $\mathbb{E}[S_\varepsilon]$ will converge asymptotically to the true value of $\mathbb{E}[S_\varepsilon]$, which is true for both Hoeffding's inequality and the betting-based algorithms in `confseq`).

Even so, the guarantee provided by Theorem 4.6 is somewhat weaker than that of Proposition 4.4, because the $1-\beta$ chance only holds for random $X \sim p(X)$ and may not hold after conditioning on additional information (e.g. the event A, which can be any function of the output of the model). To give some intuition of why this occurs, suppose we perturb a calibrated model with a tiny amount of per-input noise, e.g. $\hat{p}^\theta_{Y|X}(y|X) = p(y|\Phi(X)) + \eta(X)$. Even if $\eta(X)$ is very small, conditioning on $\hat{p}^\theta_{Y|X}(y|X)$ may then be enough to identify X itself, and if there is a single such X that is outside of the range given by Proposition 4.4, the stronger statement will no longer hold. One way to circumvent this in principle would be to explicitly bin the outputs of a model to a finite number of outputs, similar to the method proposed by Kumar et al. (2019); it would then be possible to construct a separate bound for each bin. Effectively, this would mean that we enumerate all of the events A that we care about in advance, and then apply Theorem 4.6 separately to each subset of the dataset. (Note that neither bound holds conditioned on X itself, because once X is observed then either $p(Y|X)$ is in the interval or it is not, so the conditional probability is either 0 or 1, not $1-\beta$. This is why the event in Proposition 4.4 must be conditionally independent of X given the output of the model.)

An interesting point of comparison is Theorem 1 of Barber (2020), which states that any distribution-free $(1-\alpha)$-confidence interval for the probability $p(Y=1|X)$ must also be a $(1-\alpha)$-confidence interval for any random variable $Z \in [0,1]$ for which $\mathbb{E}[Z|X] = p(Y|X)$, as long as the interval was constructed using only one sample $Y \sim p(Y|X)$ for each X (and as long as Z is conditionally independent of the interval-construction procedure given X). In particular, if we choose $Z = Y$, this means that any distribution-free $(1-\alpha)$-confidence interval must contain 0 or 1 with probability at least $(1-\alpha)$, and so the interval cannot precisely identify $p(Y|X)$ if $p(Y|X)$ is bounded away from 0 and 1.

We can roughly interpret Barber's theorem as stating that distribution-free confidence intervals constructed using *one* Y for each X can only effectively estimate the *first* moment of $p(Y|X)$, and must be wide enough to contain the worst-case variable Z with the correct expected value. Our Theorem 4.6, on the other hand, uses *two* Ys for each X, and converges to a bound based on the first *two* moments (mean and variance). Moreover, Theorem 3.2 suggests that two samples may in fact be necessary to estimate the second moment in this manner. We conjecture that this is a general constraint for distribution-free confidence intervals based on samples: if we are allowed to use k samples $Y_1, \ldots, Y_k \sim p(Y|X)$ for each X, it seems likely that only the first k moments can be identified in a distribution-free way, and thus that our confidence intervals must be wide enough to contain any random variable Z with the same first k moments as the true probability $p(Y|X)$ conditioned on the output of our model or algorithm. If true, this would suggest that we can't do much better than Theorem 4.6 with only two samples of Y for each X, unless we are willing to make distributional assumptions about the form of $p(Y|X)$, but we might be able to do better with more than two samples.

We also note that if you want a one-sided bound instead of a two-sided bound, it is possible to do better than Chebyshev's inequality by instead using Cantelli's inequality (Cantelli, 1929). This inequality was used to prove Theorem 4.5 and Proposition D.1, and it could likely be generalized to apply without assuming calibration using a similar technique to the proof of

[5] https://github.com/gostevehoward/confseq

Theorem 4.6.

E. Properties of Calibrated Models of Pairs

In this section we derive some properties that any calibrated model $\hat{p}_\theta(Y_1, Y_2|X)$ must satisfy, which can be useful when designing neural network architectures for pair prediction.

Proposition E.1. *Suppose $|\mathcal{Y}| = K$, and order it as $\mathcal{Y} = \{v_1, \dots, v_K\}$. If $\hat{p}_\theta(Y_1, Y_2|X)$ is a perfectly-calibrated predictor of outcomes $(Y_1, Y_2) \in \mathcal{Y} \times \mathcal{Y}$, then*

(i) $\hat{p}_\theta(y_1, y_2|x)$ is a proper probability distribution, i.e. $\hat{p}_\theta(y_1, y_2|x) \geq 0$ for all $x \in \mathcal{X}, y_1, y_2 \in \mathcal{Y}$ and $\sum_{y_1,y_2} \hat{p}_\theta(y_1, y_2|x) = 1$ for all $x \in \mathcal{X}$,

(ii) $\hat{p}_\theta(y_1, y_2|x)$ is symmetric, i.e. $\hat{p}_\theta(Y_1 = y_1, Y_2 = y_2|x) = \hat{p}_\theta(Y_1 = y_2, Y_2 = y_1|x)$,

(iii) The joint probability matrix $\hat{P}^{[x]} \in \mathbb{R}^{K\times K}$ given by $\hat{P}^{[x]}_{ij} = \hat{p}_\theta(Y_1 = v_i, Y_2 = v_j|x)$ is positive semidefinite for each $x \in \mathcal{X}$.

Proof. Since $\hat{p}_\theta$ is perfectly calibrated, there exists a grouping function Φ such that

$$\begin{aligned}\hat{p}_\theta(Y_1 = y_1, Y_2 = y_2|X = x) &= \mathbb{E}\big[p(Y_1 = y_1, Y_2 = y_2|X) \;\big|\; \Phi(X) = \Phi(x)\big] \\ &= \mathbb{E}\big[p(Y = y_1|X)p(Y = y_2|X) \;\big|\; \Phi(X) = \Phi(x)\big],\end{aligned}$$

which implies properties (i) and (ii).

If we let $\boldsymbol{p}^{[x]} \in \mathbb{R}^K$ be the vector such that $\boldsymbol{p}^{[x]}_k = p(Y = v_k|X = x)$, we can write this in matrix form as

$$\hat{P}^{[x]} = \mathbb{E}\Big[\boldsymbol{p}^{[x]}\big(\boldsymbol{p}^{[x]}\big)^T \;\Big|\; \Phi(X) = \Phi(x)\Big].$$

For any $\boldsymbol{v} \in \mathbb{R}^K$, we must then have

$$\boldsymbol{v}^T\hat{P}^{[x]}\boldsymbol{v} = \mathbb{E}\Big[\boldsymbol{v}^T\boldsymbol{p}^{[x]}\big(\boldsymbol{p}^{[x]}\big)^T\boldsymbol{v} \;\Big|\; \Phi(X) = \Phi(x)\Big] = \mathbb{E}\Big[\big(\boldsymbol{v}^T\boldsymbol{p}^{[x]}\big)^2 \;\Big|\; \Phi(X) = \Phi(x)\Big] \geq 0,$$

so $\hat{P}^{[x]}$ is positive semidefinite. □

We note that a nonnegative matrix satisfying (ii) and (iii) is known as a "doubly nonnegative" matrix, and if $p(Y|X)$ takes only finitely many values the matrix $P^{[x]}$ will also be "completely positive" (i.e. factorizable as $P^{[x]} = B^TB$ where B is entrywise nonnegative) (Berman & Shaked-Monderer, 2003).

If Y is a binary outcome, we can characterize the space of calibrated predictors even more precisely:

Proposition E.2. *If $\hat{p}_\theta(Y_1, Y_2|X)$ is a perfectly-calibrated predictor of paired binary outcomes $(Y_1, Y_2) \in \{0, 1\} \times \{0, 1\}$, then the matrix $\hat{P}^{[x]}$ can be written in the form*

$$\hat{P}^{[x]} = \rho(x)\begin{bmatrix}1-\mu(x) & 0\\ 0 & \mu(x)\end{bmatrix} + (1-\rho(x))\begin{bmatrix}(1-\mu(x))^2 & \mu(x)(1-\mu(x))\\ \mu(x)(1-\mu(x)) & \mu(x)^2\end{bmatrix} \tag{3}$$

for some $\mu : \mathcal{X} \to [0, 1]$ and $\rho : \mathcal{X} \to [0, 1]$.

Proof. Fix a particular x. By Proposition E.1 we know we can write

$$\hat{P}^{[x]} = \begin{bmatrix}a & b\\ b & c\end{bmatrix}$$

for some nonnegative $a, b, c \in \mathbb{R}$ such that $a + 2b + c = 1$. Choose

$$\mu(x) = b + c, \qquad \rho(x) = \frac{ac - b^2}{(a + b)(b + c)} = 1 - \frac{b}{\mu(1-\mu)}. \tag{4}$$

Substituting shows that $\hat{P}^{[x]}$ can then be expressed as Equation (3). $0 \leq \mu(x) \leq 1$ because b and c are nonnegative and sum to at most 1. $\rho(x) \leq 1$ because $ac - b^2 \leq ac \leq (a + b)(c + d)$. Finally, since $\hat{P}^{[x]}$ must be positive semidefinite, the determinant $|\hat{P}^{[x]}| = ac - b^2$ must be nonnegative, so $\rho(x) \geq 0$. □

Note that μ is the predicted probability of $Y = 1$, and ρ is the predicted correlation between Y_1 and Y_2. (In fact, ρ is exactly the Pearson correlation coefficient of Y_1 and Y_2 given $\Phi(X)$, also referred to as the "Phi coefficient" (Chedzoy, 2005).) Given this parameterization, we can efficiently compute

$$\hat{p}_\theta(Y = 1|X = x) = \mu(x), \qquad v_\theta^{\text{CHEAT}}(Y = 1|X = x) = \kappa(x)\mu(x)(1 - \mu(x)).$$

Neural network architectures for $\mathcal{Y} = \{0, 1\}$: When we know Y is a binary outcome, we suggest parameterizing the output head of a pair predictor $\hat{p}_\theta(y_1, y_2|x)$ using Equation (3). Specifically, we can parameterize our model to produce two-dimensional vectors $h_\theta : \mathcal{X} \to \mathbb{R}^2$, then set $\phi(x) = \sigma(h_\theta(x)[0])$, $\rho(x) = \sigma(h_\theta(x)[1])$, where σ is the logistic sigmoid function.

Neural network architectures for enumerable $\mathcal{Y} = \{0, 1, \ldots, K\}$: For classification tasks, where $\mathcal{Y}$ is a finite (and "reasonably-sized") set of classes, we suggest using the properties in Proposition E.1 to design the architecture. In particular, we can enforce property (i) by applying the `softmax` operation across the set of $\mathcal{Y} \times \mathcal{Y}$ possible outputs, and enforce property (ii) by constraining the output layer to output a symmetric matrix of logits $\mathbb{R}^{\mathcal{Y}\times\mathcal{Y}}$ before applying the softmax operation.

We are not aware of a simple method for strictly enforcing property (iii) as part of the architecture while simultaneously ensuring that property (i) holds. However, empirically we observe that violations of property (iii) can lead to unreasonable negative variance estimates. We thus suggest computing the eigenvalues of the post-softmax matrix $\hat{P}^{[x]}$ and adding a regularization penalty to negative eigenvalues, e.g.

$$\mathcal{L}_{\text{regularized}}(x, y_1, y_2) = -\log \hat{p}_\theta(y_1, y_2|x) + \alpha \sum_{i=1}^{K} \max\{0, \lambda_i(x)\}^2$$

where $\lambda_i(x)$ is the ith eigenvalue of $\hat{P}^{[x]}$. (Since this regularization penalty only applies to negative eigenvalues, and a calibrated model should never produce negative eigenvalues, this regularization penalty should not change the optimal calibrated solution if one exists.)

Neural network architectures for sequential or exponentially-large $\mathcal{Y}$: When $\mathcal{Y}$ is an exponentially large set, such as the set of all sequences, it may be intractable to enforce either condition (ii) or condition (iii) of Proposition E.1. For our experiments, we settled on only enforcing property (i) by concatenating the two outputs Y_1 and Y_2 together. We found that padding them to a constant length improved performance by ensuring that Y_1 and Y_2 each have consistent positional embeddings, because othewise the positional shift in Y_2 can make it harder to predict Y_2 than Y_1 and thus introduce additional noise into the confidence metric. We believe adjusting the architecture for sequence models to enforce (or encourage) it to satisfy properties (ii) and (iii) is an exciting area for future work.

F. Details of Experimental Results

F.1. One-Dimensional Binary Regression (Figure 3)

F.1.1. Data Distribution

We choose $p(X)$ as a standard normal random variable $\mathcal{N}(0, 1)$, and define $p(Y|X)$ as a Bernoulli distribution with

$$\begin{aligned} p(Y = 1|X = x) &= \frac{0.98\, u(x) + 1}{2}, \\ u(x) &= 0.6 \cos(v(x)) + 0.4 \cos(4.2x), \\ v(x) &= \text{sign}(x) \cdot (120|x| - 112w(|x|) - 0.0635). \\ w(z) &= 0.2 \log\Big(1 + \exp\big((z - 1.0)/.2\big)\Big) \end{aligned}$$

This function was chosen to have higher-frequency variation near $x = 0$ with a lower-frequency component throughout.

We construct a dataset of 25,000 samples of X, each of which have two corresponding samples $Y_1, Y_2 \sim_{i.i.d.} p(Y|X)$, for a total of 50,000 Ys.

F.1.2. Architectures and Training Details

For the NN Ensemble, Evidential NN, and Cheat-corrected NN models, we use a small MLP/LayerNorm/Residual architecture inspired by the MLP blocks in a Transformer (Vaswani et al., 2017), with the following form:

Algorithm 2 NN architecture for 1D Binary Regression

Input: value $x \in \mathbb{R}$, output dimension d
Input layer:
$\boldsymbol{v}^{(0)} := \boldsymbol{w}^{(0,a)} \odot (x \cdot \mathbf{1} + \boldsymbol{b}^{(0,a)})$ where $\boldsymbol{w}^{(0,a)} \in \mathbb{R}^{512}, \boldsymbol{b}^{(0,a)} \in \mathbb{R}^{512}$
$\boldsymbol{r}^{(0)} := \boldsymbol{W}^{(0,b)}\text{relu}(\boldsymbol{v}^{(0)}) + \boldsymbol{b}^{(1,b)}$ where $\boldsymbol{W}^{(0,b)} \in \mathbb{R}^{128\times512}, \boldsymbol{b}^{(0,b)} \in \mathbb{R}^{128}$
for $i = 1$ **to** 3 **do**
  Residual block:
  $\boldsymbol{u}^{(i)} := \text{LayerNorm}^{(i)}(\boldsymbol{v}^{(i-1)})$ with learnable scale and shift (Ba et al., 2016)
  $\boldsymbol{v}^{(i)} := \boldsymbol{W}^{(i,a)}\boldsymbol{u}^{(i)} + \boldsymbol{b}^{(i,a)}$ where $\boldsymbol{W}^{(i,a)} \in \mathbb{R}^{512\times128}, \boldsymbol{b}^{(i,a)} \in \mathbb{R}^{512}$
  $\boldsymbol{r}^{(i)} := \boldsymbol{r}^{(i-1)} + \boldsymbol{W}^{(i,b)}\text{relu}(\boldsymbol{v}^{(i)}) + \boldsymbol{b}^{(i,b)}$ where $\boldsymbol{W}^{(i,b)} \in \mathbb{R}^{128\times512}, \boldsymbol{b}^{(i,b)} \in \mathbb{R}^{128}$
end for
Output head:
$\boldsymbol{u}^{(4)} := \text{LayerNorm}^{(4)}(\boldsymbol{v}^{(3)})$ with learnable scale and shift
$\boldsymbol{v}^{(4)} := \boldsymbol{W}^{(4,a)}\boldsymbol{u}^{(4)} + \boldsymbol{b}^{(4,a)}$ where $\boldsymbol{W}^{(4,a)} \in \mathbb{R}^{512\times128}, \boldsymbol{b}^{(4,a)} \in \mathbb{R}^{512}$
$\boldsymbol{o}^{(4)} := \boldsymbol{W}^{(4,b)}\text{relu}(\boldsymbol{v}^{(4)}) + \boldsymbol{b}^{(4,b)}$ where $\boldsymbol{W}^{(4,b)} \in \mathbb{R}^{d\times512}, \boldsymbol{b}^{(4,b)} \in \mathbb{R}^{d}$
Return $\boldsymbol{o}^{(4)}$

NN Ensemble: We randomly initialize 8 copies of the architecture with output dimension $d = 1$, then train each for 10,000 training iterations with a batch size of 512, randomly selecting (X, Y_1, Y_2) triples from the 25,000 training examples. For each example (x, y_1, y_2) we use the loss

$$\mathcal{L}_{\text{NN}}(x, y_1, y_2, \theta_i) = \frac{1}{2}\sum_{i=1}^{2} -\log \hat{p}_{\theta_i}(Y = y_i|X = x)$$

where $\hat{p}_{\theta_i}(Y = 1|X = x) = \sigma(h_{\theta_i}(x))$, σ is the logistic sigmoid function, and h_{θ_i} is the network defined in Algorithm 2. We use the AdamW optimizer (Loshchilov & Hutter, 2017) with a 100-step warmup to a 0.002 learning rate, followed by cosine decay.

Evidential NN: Following (Sensoy et al., 2018), we set the output dimension to $d = 2$ and interpret $\boldsymbol{\alpha}(x) = 1 + \texttt{softplus}(h_\theta(x))$ as the parameters of a 2-class Dirichlet distribution. (We use `softplus` rather than `relu` to stabilize learning, since otherwise we observed that output units would "die" and produce bad estimates.)

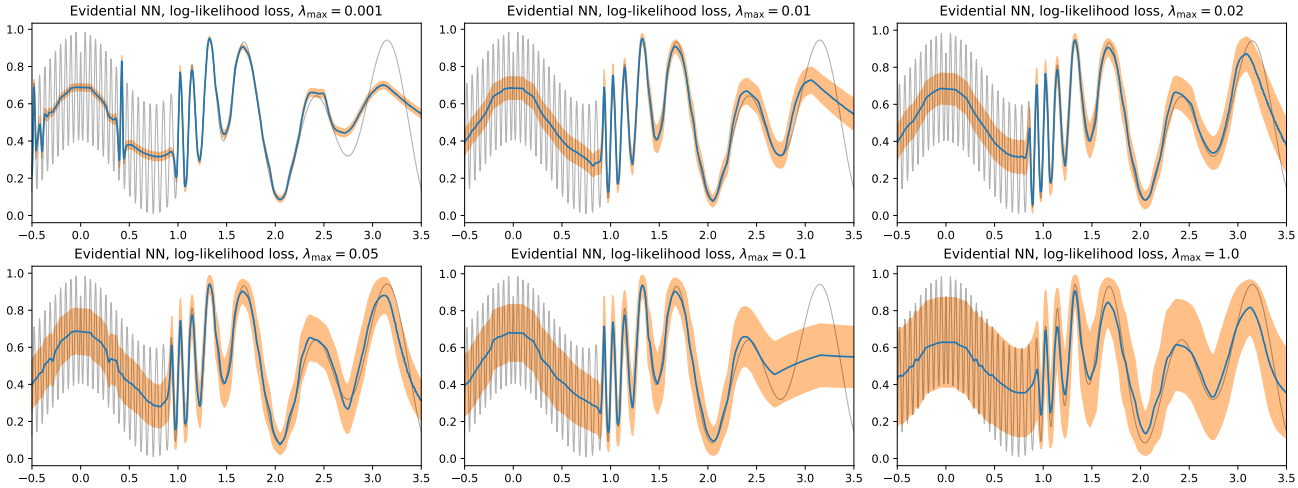

Figure 16. Visualization of the dependence of the evidential deep learning technique (Sensoy et al., 2018) on the final regularization strength $\lambda_{\max}$. Sensoy et al. (2018) recommend setting $\lambda_{\max} = 1$, which leads to high-uncertainty predictions even in regions that the model can fit well. We are unable to find any $\lambda_{\max}$ value that allows the model to identify underfitting.

We then apply the regularized cross-entropy loss described by Sensoy et al. (2018):

$$\begin{aligned}\mathcal{L}_{\text{EDL}}(x, y_1, y_2, \theta) &= \frac{1}{2}\sum_{i=1}^{2}\left[\mathbb{E}_{q\sim\text{Dirichlet}(\boldsymbol{\alpha}(x))}[-\log q(y_i)] + \lambda D_{KL}\left(\text{Dirichlet}(\tilde{\boldsymbol{\alpha}}(x, y_i)) \parallel \text{Dirichlet}([1,1])\right)\right] \\ &= \frac{1}{2}\sum_{i=1}^{2}\left[\psi(\mathbf{1}^T\boldsymbol{\alpha}(x)) - \psi(\boldsymbol{e}_{y_i}^T\boldsymbol{\alpha}(x)) + \lambda D_{KL}\left(\text{Dirichlet}(\tilde{\boldsymbol{\alpha}}(x, y_i)) \parallel \text{Dirichlet}([1,1])\right)\right],\end{aligned}$$

where $\boldsymbol{\alpha}(x)$ is the two-dimensional vector of model outputs, $\boldsymbol{e}_{y_i}$ is a one-hot indicator vector (either [1,0] or [0,1] depending on y_i), ψ is the digamma function, and $\tilde{\boldsymbol{\alpha}}(x, y_i) = \boldsymbol{e}_{y_i} + (1 - \boldsymbol{e}_{y_i}) \odot \boldsymbol{\alpha}(x)$ is a vector where the Dirichlet parameter for the correct label has been replaced with 1.

Similar to the NN ensemble, we train the model for 10,000 training iterations with a batch size of 512, randomly selecting (X, Y_1, Y_2) triples from the 25,000 training examples, and use the AdamW optimizer (Loshchilov & Hutter, 2017) with a 100-step warmup to a 0.002 learning rate, followed by cosine decay. We interpolate λ from 0 to 1 over 5,000 training steps, based on the recommended values for λ in (Sensoy et al., 2018).

Sensoy et al. (2018) suggest using the magnitude of $\boldsymbol{\alpha}(x)$ as a measurement of evidence, with the uncertainty corresponding to the value $2/\mathbf{1}^T\boldsymbol{\alpha}(x)$ (for two classes). However, in order to treat Evidential Deep Learning in the same way as other uncertainty estimates, we instead use the variance of the predicted probability under Dirichlet$(\boldsymbol{\alpha}(x))$ as our measurement of uncertainty.

In Figure 3, we plot the mean probability $\hat{p}(x)$ under the distribution Dirichlet$(\boldsymbol{\alpha}(x))$, as well as the variance $\hat{v}(x) = \frac{\hat{p}(x)(1-\hat{p}(x))}{S(x)+1}$. We additionally trained variants of EDL with different maximum values for λ, and found that the magnitude of the variance estimate is highly sensitive to this, as we demonstrate in Figure 16. The model in Figure 3 uses $\lambda = 1.0$. (We also tried applying EDL with the MSE loss, as suggested by Sensoy et al. (2018), but saw roughly identical behavior.)

Cheat-corrected NN: We parameterize our version of the architecture by setting $d = 2$ and applying the decomposition in Equation (3) of Appendix E. We then train it to predict pairs using the loss

$$\mathcal{L}_{\text{CHEAT}}(x, y_1, y_2, \theta) = -\log \hat{p}_\theta(Y_1 = y_1, Y_2 = y_2 | X = x).$$

We again train for 10,000 training iterations with a batch size of 512, randomly selecting (X, Y_1, Y_2) triples from the 25,000 training examples, and use the AdamW optimizer (Loshchilov & Hutter, 2017) with a 100-step warmup to a 0.002 learning rate, followed by cosine decay. We then compute $\hat{p}_\theta(Y = 1 | X = x)$ and $v_\theta^{\text{CHEAT}}(Y = 1 | X = x)$ as described in Appendix E.

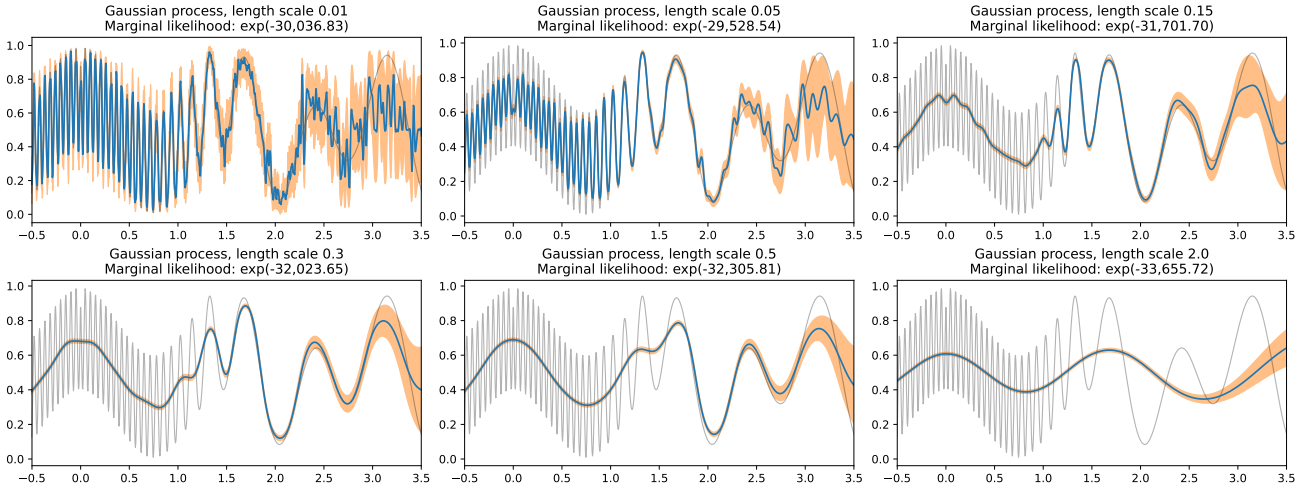

Figure 17. Visualization of the dependence of the Gaussian process inducing-points classifier on the length scale, and the estimated marginal likelihood of the dataset points under each prior.

Gaussian process classifier: For our Gaussian process experiment, we follow a standard discriminative Gaussian process classifier setup (Rasmussen & Williams, 2005): we impose a Gaussian process prior over a latent "logit" function $f : \mathbb{R} \to \mathbb{R}$, then feed it through the logistic sigmoid transformation to obtain a conditional likelihood

$$p(Y = 1|f, x) = \sigma(f(x)) = \frac{1}{1 + \exp(-f(x))}.$$

Given an observed dataset $\mathcal{D} = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$ of (X, Y) pairs, we can then approximately compute the posterior distribution

$$p\left(f \middle| \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}\right) \propto p(f) \prod_{i=1}^{N} p(y^{(i)}|f, x^{(i)})$$

and use it to compute the posterior mean and variance for a new data point x:

$$p(Y = 1|x, \mathcal{D}) = \int_f \sigma(f(x)) p(f|\mathcal{D})\, df,$$
$$\mathrm{Var}\left[p(Y = 1|f, x)\middle|\mathcal{D}\right] = \int_f \sigma(f(x))^2 p(f|\mathcal{D})\, df - p(Y = 1|x, \mathcal{D})^2.$$

For this task, we select a rational-quadratic kernel with standard deviation 2.0, mixture parameter 1.0, and length scale 0.15:

$$\mathrm{Cov}\big(f(a), f(b)\big) = 2.0^2 \left(1 + \frac{(b-a)^2}{2 \times 0.15^2}\right)^{-1}$$

Our training set includes 50,000 Y samples, so computing an analytic posterior over f is computationally difficult. We instead use a variational approximation using inducing points, following (Hensman et al., 2015): we choose K inducing points $z^{(1)}, \ldots, z^{(K)}$, let $u = \{f(z^{(k)})\}_{k=1}^{K}$ be the latent function values for those points, then impose an approximate posterior

$$q(f) = \int_u p(f|u) q(u)\, du,$$

which can be used to construct an evidence lower bound on the likelihood

$$\log p(\mathcal{D}) \geq \mathbb{E}_{q(f)} \left[\sum_i \log p(y^{(i)}|f, x^{(i)})\right] + D_{KL}(q(u)\|p(u)).$$

We select $K = 512$ inducing points evenly spaced between -4 and 4, parameterize $q(u)$ as a Cholesky-factorized multivariate normal distribution $q(u) = \mathcal{N}(u; \mu, LL^T)$ where $\mu \in \mathbb{R}^{512}$, $L \in \mathbb{R}^{512\times512}$, and use the Cholesky factorization to analytically compute the KL divergence. (For numerical stability purposes, we add 0.001 to the diagonal of the prior covariance matrix.) We then maximize the evidence lower bound above, approximating the expectation by subsampling 512 (x, y_1, y_2) triples per iteration and using Gauss-Hermite quadrature over the distribution $q(f(x^{(i)})|u)$; we treat the two samples y_1 and y_2 for each x as independent observations (x, y_1), (x, y_2). We optimize μ and L for 20,000 training iterations using stochastic gradient descent, with a maximum learning rate of 0.05, 100 steps of warmup, and a cosine decay schedule, although we observe that the approximate posterior converges within about half of that time.

We note that the degree of misspecification varies based on the length scale, as shown in Figure 17, because the true function does not have a consistent length scale and was not chosen from the prior. If we know in advance which length scales to try, the marginal likelihood estimates (our bound on $p(\mathcal{D})$) may allow us to identify the best-fitting model (in this case, the version with length scale 0.05, although it is imperfect). However, the estimates of the variance of f does not provide a good estimate of pointwise misspecification; we chose to use length scale 0.15 in Figure 3 to emphasize this. (In real world settings, misspecification would likely be much harder to detect or correct, especially without a thorough hyperparameter sweep.) We also note that the marginal likelihoods of each approach are roughly of the same order; similar-looking data could plausibly have been generated by even the misspecified models because the data itself consists of binary outputs.

An expanded version of each of the parts of Figure 3 is shown in Figure 18.

F.1.3. Additional results

Figure 19 shows a reliability diagram for first-order calibration for the task in Figure 3, demonstrating that all methods are close to first-order calibrated on this task. Figure 20 shows a similar plot for second-order calibration, which indicates that our method is indeed better second-order calibrated. Each point in these figures was computed by aggregating over 20 equal-probability-mass bins; perfect calibration would correspond to a diagonal line with slope 1.

To show that these results are not specific to the particular sinusoidal function we chose, we also present results for a randomly-selected piecewise linear function, shown in Figure 21. Similar to the sinusoidal function in Figure 3, the ensemble and Gaussian process methods tend to overestimate their confidence for high-probability inputs around $x = 0$. Our technique gives a better estimate for common x, although it is overconfident for $x < -3$. Overall, it is better second-order calibrated and similarly first-order calibrated, as shown in Figures 22 and 23.

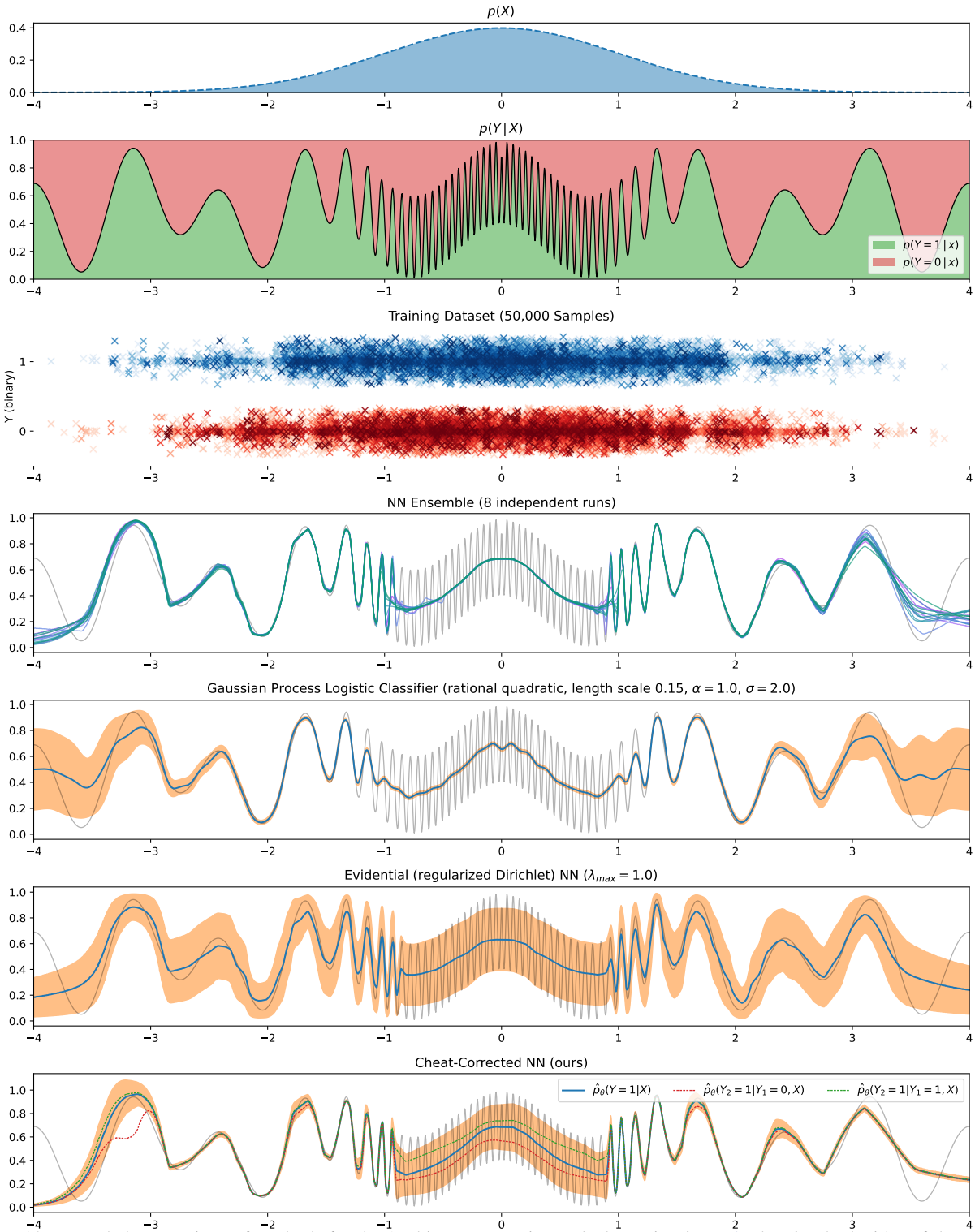

Figure 18. Expanded comparison of methods for the 1D binary regression task shown in Figure 3, showing both sides of the normal distribution $p(X)$. Note that our method is the best at distinguishing underfitting (near 0) from accurate estimation (1 to 3). Our technique does not directly do out-of-distribution detection, so this model's uncertainty estimates are overconfident outside the range of the dataset ($x > 3$); this could be fixed by combining our method (which detects underfitting) with another method that improves out-of-distribution calibration (as we did with the Cheat-SNGP model in the CIFAR-10H task).

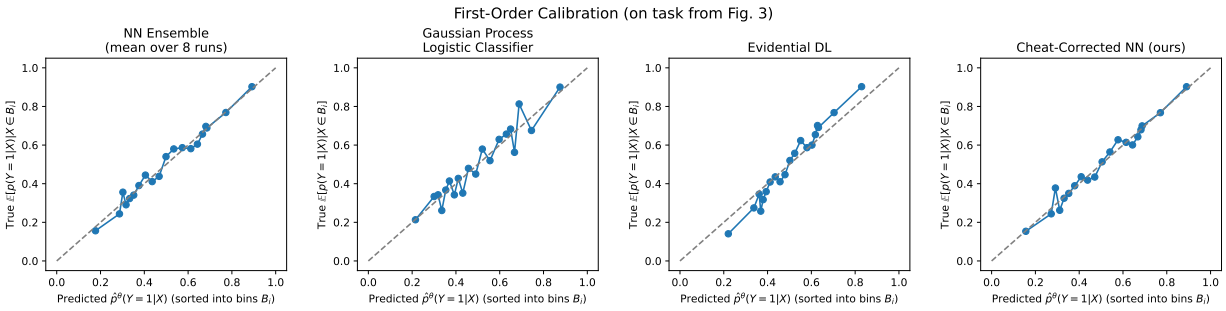

Figure 19. First-order calibration reliability diagram for the task in Figure 3. All methods are close to first-order calibrated on this task.

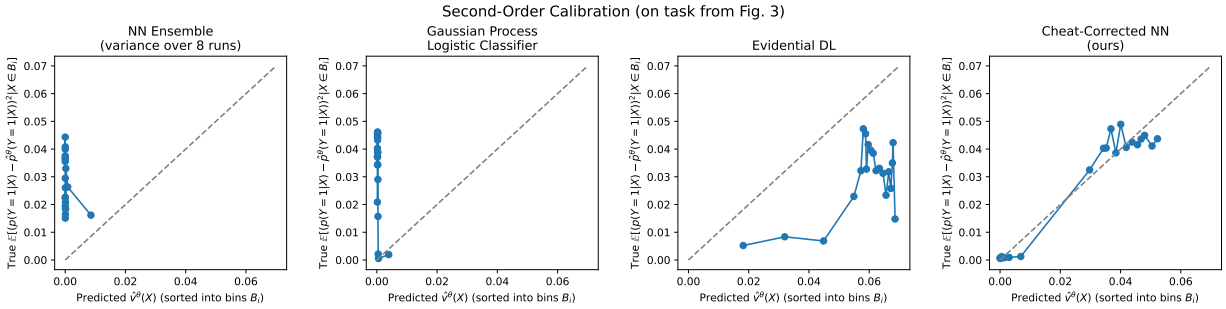

Figure 20. Second-order calibration reliability diagram for the task in Figure 3. Our method is better second-order-calibrated than the baseline approaches.

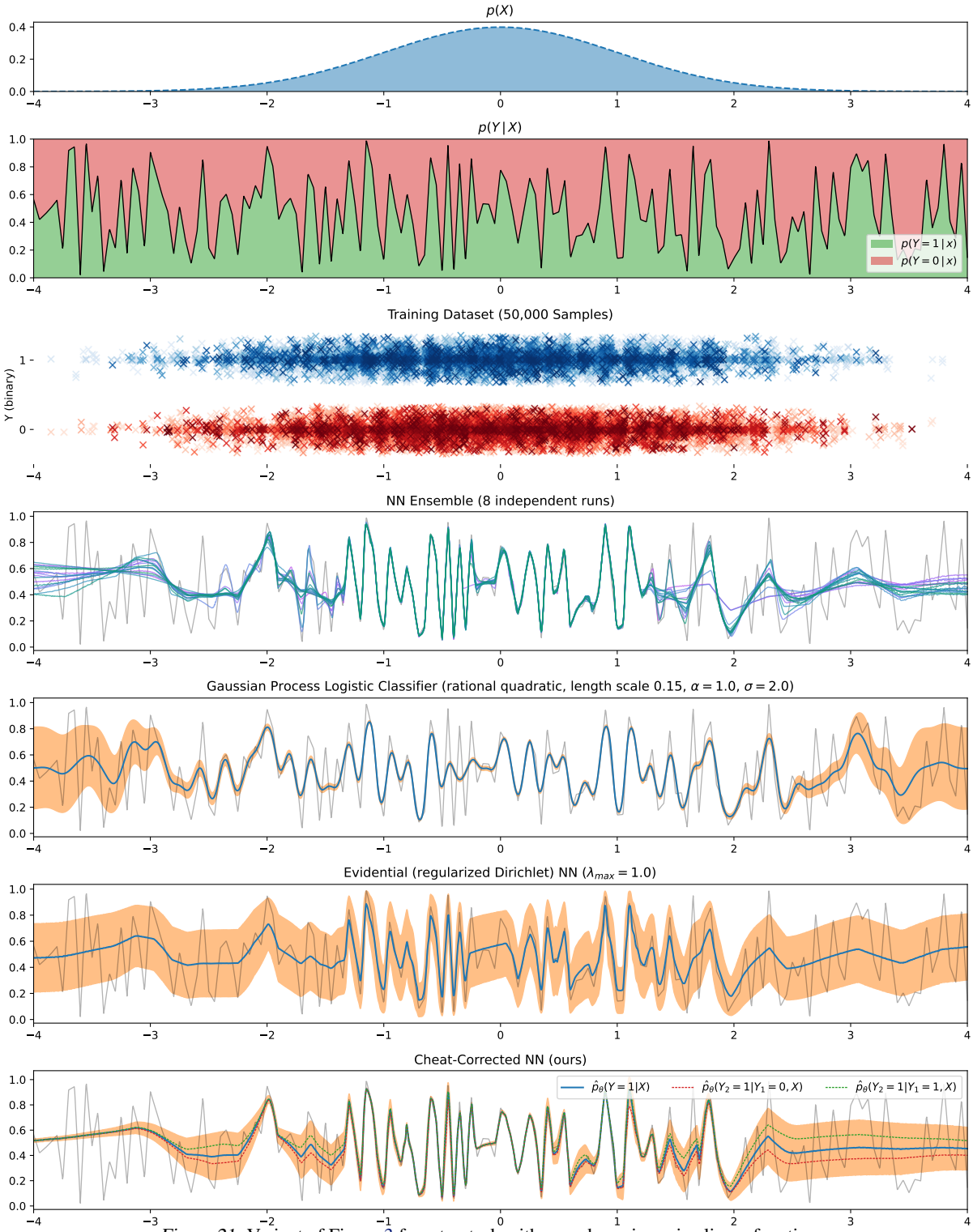

Figure 21. Variant of Figure 3 for a toy task with a random piecewise-linear function.

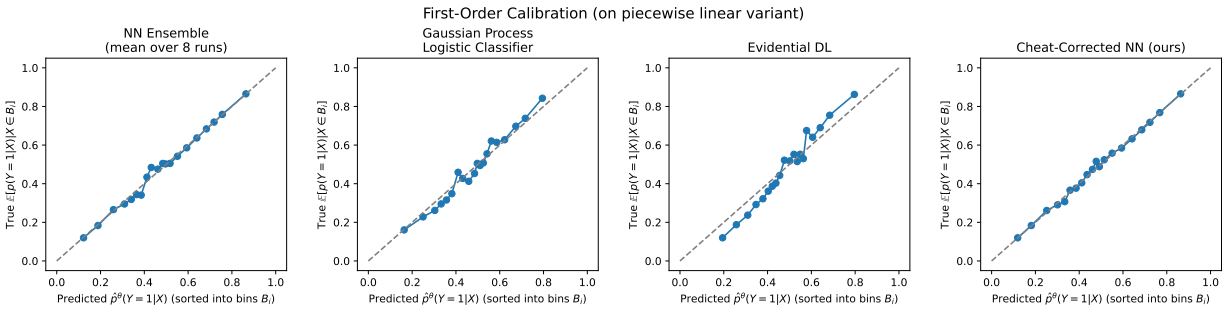

Figure 22. First-order calibration reliability diagram for the task in Figure 21. All methods are close to first-order calibrated on this task.

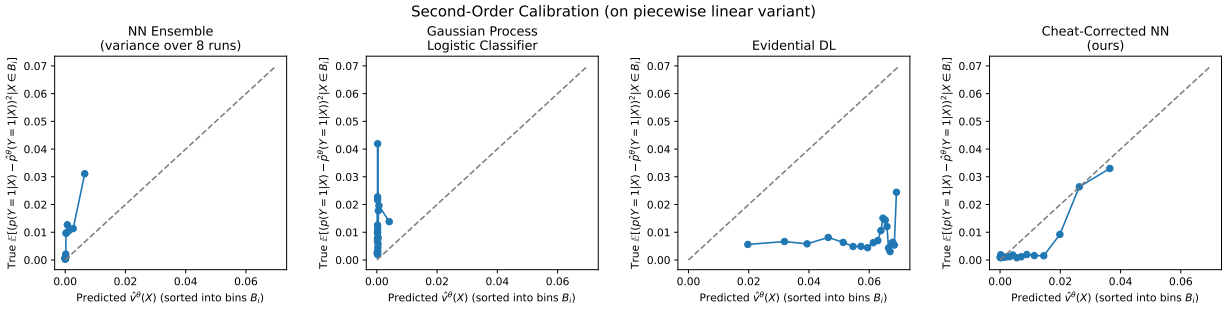

Figure 23. Second-order calibration reliability diagram for the task in Figure 21. Our method is better second-order-calibrated than the baseline approaches.

F.2. CIFAR-10H

F.2.1. Training and hyperparameter tuning strategy

Due to the small size of CIFAR-10H (Peterson et al., 2019), we split the dataset into multiple parts and combine it with CIFAR-10N (Wei et al., 2021):

- We use the 50,000 images in CIFAR-10N as a *pretraining set*; this corresponds to the training set of the original CIFAR-10 dataset. Each image has three labels from random annotators; we select two from these randomly at each training step. (Note that the distribution of aleatoric noise in CIFAR-10N is not exactly the same as the aleatoric noise in CIFAR-10H, likely due to being labeled at different times by different annotators, under the direction of different authors.)
- We use the first 3,000 images in CIFAR-10H as *finetuning set 1*. Each image has at least 50 annotations; we select two from these randomly at each training step.
- We use the next 2,000 images in CIFAR-10H as our *validation set*. We use the 50 labels per image as a proxy for the true distribution $p(Y|X = x)$, and take the KL divergence between this empirical distribution and the model's predictions as our hyperparameter tuning objective. After hyperparameter tuning, we reuse this set of images as *finetuning set 2*, combining it with the first 3,000 images in CIFAR-10H to form a 5,000-image set.
- Finally, we use the last 5,000 images in CIFAR-10H as our *test set*, and use it to evaluate our final metrics.

We apply the AugMix augmentation strategy (Hendrycks et al., 2019) when sampling examples from the pretraining and finetuning sets, to improve robustness of all methods. We also normalize pixel values based on the mean and standard deviation of pixels across the full CIFAR-10 dataset, following the implementation in `uncertainty_baselines`.

We train each method using the AdamW optimizer (Loshchilov & Hutter, 2017) with batch size 512. We divide our training and hyperparameter tuning into the following phases:

- *Phase 1: Pretraining hyperparameter sweep.* We train each method on the CIFAR-10N pretraining set for 50 epochs. We perform a random search over learning rate and weight decay strength with 250 trials: we choose learning rate logarithmically spaced between 10^{-5} and 5×10^{-3}, and we either sample weight decay uniformly between 0.05 and 0.5, or logarithmically between 10^{-6} and 0.05, since different tasks and methods may benefit from either very strong or very weak weight decay. We use a linear warmup for the learning rate during the first epoch, then use cosine weight decay.
- *Phase 2: Pretraining extended.* We take the best-performing hyperparameters from phase 1, as judged by validation KL divergence, and retrain that configuration from scratch for both 500 and 200 epochs.
- *Phase 3: Fine-tuning hyperparameter sweep.* We train each method on our CIFAR-10H finetuning set 1. We perform a random search over learning rate and weight decay, as in phase 1, and also randomly search over the number of epochs to use, either 30, 50, 100, or 200. We initialize the parameters from either of the checkpoints from Phase 2, using 250 trials in the random sweep for each checkpoint. (Effectively, we do 500 trials, where we are also tuning the number of epochs of pretraining.)
- *Phase 4: Expanded fine-tuning.* We take the best-performing configuration from Phase 3, again judged by validation KL divergence. We then reset to the checkpoint from Phase 2 (depending on the Phase 3 configuration), and train it on the combination of of finetuning set 1 and finetuning set 2 (the validation set), so that we maximize the amount of finetuning data.
- *Phase 5: Evaluation.* We take the resulting model from Phase 4 and evaluate our metrics on our test set (the second half of CIFAR-10H).

The best-performing hyperparameters for each method are shown in Table 2. Interestingly, we found that very strong weight decay was the most effective during pretraining for all models, perhaps because of the relatively small dataset and large number of epochs.

Table 2. Best-performing hyperparameters for each model architecture, based on our hyperparameter sweeps.

	PRETRAIN (CIFAR-10N)			FINETUNE (CIFAR-10H)		
METHOD	LEARNING RATE	WEIGHT DECAY	EPOCHS	LEARNING RATE	WEIGHT DECAY	EPOCHS
NAIVE NN / NN ENSEMBLE	3.799E-03	3.656E-01	50	9.071E-05	2.363E-01	50
EVIDENTIAL DL	7.465E-04	2.950E-01	200	1.455E-04	2.146E-03	200
SNGP COV.	1.221E-03	4.742E-01	200	6.032E-05	1.457E-01	100
EPINET	1.327E-03	4.513E-01	50	7.345E-05	2.954E-01	50
CHEAT NN	1.113E-03	4.835E-01	50	4.265E-05	1.865E-01	50
CHEAT SNGP	1.687E-03	4.411E-01	200	4.972E-05	7.621E-05	30

F.2.2. MODEL ARCHITECTURES

We implement all of our models and baselines using the `uncertainty_baselines` Python library (Nado et al., 2021), building on TensorFlow (Abadi et al., 2015) and Keras (Chollet et al., 2015). All methods are based on the wide ResNet architecture (Zagoruyko & Komodakis, 2016) as implemented in `uncertainty_baselines`, with depth 28 and a width multiplier of 10.

Naive NN: We configure the wide ResNet with 10 output classes and a softmax output layer, and train it using the ordinary cross entropy (negative log likelihood) loss.

At test time, we use $\hat{V}^\theta(y|x) = \hat{p}^\theta_{Y|X}(y|x)(1 - \hat{p}^\theta_{Y|X}(y|x))$ as an estimate of variance. This corresponds to the assumption that all noise is epistemic, and would make sense if we knew Y was a deterministic function of X. (However, for this task we know this is not the case, so this will overestimate uncertainty.)

NN Ensemble: We use the same configuration as for Naive NN, but train eight copies of the model. (We do not perform a separate hyperparameter tuning sweep for NN ensemble, since it would be the same as the sweep for Naive NN.)

At test time, we take the empirical mean and variance across the ensemble as our prediction and epistemic uncertainty estimates:

$$\hat{p}_{\theta_1,\ldots,\theta_8}(y|x) = \frac{1}{8}\sum_{i=1}^{8} \hat{p}_{\theta_i}(y|x)$$

$$\hat{v}_{\theta_1,\ldots,\theta_8}(y|x) = \frac{1}{7}\sum_{i=1}^{8} \left(\hat{p}^\theta_{Y|X}(y|x) - \hat{p}_{\theta_1,\ldots,\theta_8}(y|x)\right)^2$$

We divide by 7 so that our sample variance estimator is an unbiased estimate of the variance under a hypothetical infinite ensemble.

SNGP Cov.: We use the SNGP variant of a wide ResNet from `uncertainty_baselines`, which applies spectral normalization to the intermediate layers of the ResNet and replaces the normal linear output layer with a random-feature Gaussian process approximation (Liu et al., 2020). We configure it using the default configuration for CIFAR-10: 1024 orthogonal random features, a 1.0 ridge penalty, a 20 mean-field factor, and a spectral-norm bound of 6.0.

Following the training script for SNGP in `uncertainty_baselines`, we use the ordinary logits of the model during training; this roughly corresponds to using the posterior mean of the learned Gaussian process posterior. After training the model, we perform another pass over the training set to compute a Laplace approximation of the covariance matrix. To transform the mean and covariance over the logits into a mean and variance over output probabilities, we use a Monte Carlo approximation by applying softmax to each of 1000 samples, then taking the mean and variance of the resulting probability vectors.

Epinet: We augment our ResNet base network architecture with a MLP epinet head, using the implementation in the official `enn` library[6] (Osband et al., 2021), which we wrap using the `jax2tf` library in JAX (Bradbury et al., 2018). We copy the hyperparameters from the CIFAR-10 checkpoints in that repository: a 20-dimensional index vector, 50-dimensional hiddens, an epinet prior scale of 4.0, and no additional convolutional prior network. The epinet head takes the penultimate layer

[6]https://github.com/google-deepmind/enn

features from the ResNet and makes an additive contribution to the ResNet's outputs, indexed by a random input. (Note that, although the `enn` library checkpoints are for a non-wide ResNet, we instead use our wide ResNet backbone for consistency with the other baselines, and train the epinet from scratch.)

We train the base network and epinet head jointly from scratch, taking an average of the ordinary cross-entropy loss across five randomly-sampled "index" vectors, as recommended by (Osband et al., 2021). We then evaluate the epinet predictions on our test set by taking the mean and variance of the post-softmax probabilities across 1000 sampled index vectors for each input.

Evidential NN: We set up the wide ResNet architecture with 10 outputs, and convert them into parameters for a Dirichlet distribution according to $\boldsymbol{\alpha}(x) = 1 + \texttt{softplus}(h_\theta(x))$. We then apply the regularized cross-entropy loss described by Sensoy et al. (2018):

$$\begin{aligned}\mathcal{L}_{\text{EDL}}(x, y_1, y_2, \theta) &= \frac{1}{2}\sum_{i=1}^{2}\left[\mathbb{E}_{q\sim\text{Dirichlet}(\boldsymbol{\alpha}(x))}[-\log q(y_i)] + \lambda D_{KL}\big(\text{Dirichlet}(\tilde{\boldsymbol{\alpha}}(x, y_i)) \parallel \text{Dirichlet}([1,1])\big)\right] \\ &= \frac{1}{2}\sum_{i=1}^{2}\left[\psi(\mathbf{1}^T\boldsymbol{\alpha}(x)) - \psi(\boldsymbol{e}_{y_i}^T\boldsymbol{\alpha}(x)) + \lambda D_{KL}\big(\text{Dirichlet}(\tilde{\boldsymbol{\alpha}}(x, y_i)) \parallel \text{Dirichlet}([1,1])\big)\right],\end{aligned}$$

where $\boldsymbol{e}_{y_i} \in \mathbb{R}^{10}$ is a one-hot indicator vector for the correct class, ψ is the digamma function, and $\tilde{\boldsymbol{\alpha}}(x, y_i) = \boldsymbol{e}_{y_i} + (1 - \boldsymbol{e}_{y_i}) \odot \boldsymbol{\alpha}(x)$ is a vector where the Dirichlet parameter for the correct label has been replaced with 1. We interpolate λ from 0 to 1 over 10 epochs as recommended by Sensoy et al. (2018).

At test time, we compute the average probabilities and variances as

$$\begin{aligned}\hat{p}^\theta_{Y|X}(Y = y|X) &= \frac{\boldsymbol{e}_y^T\boldsymbol{\alpha}(x)}{\mathbf{1}^T\boldsymbol{\alpha}(x)}, \\ \hat{V}^\theta(Y = y|X) &= \frac{\hat{p}^\theta_{Y|X}(Y = y|X)\big(1 - \hat{p}^\theta_{Y|X}(Y = y|X)\big)}{\mathbf{1}^T\boldsymbol{\alpha}(x) + 1}.\end{aligned}$$

This is the mean and variance of the probability $q(y)$ when $q \sim \text{Dirichlet}(\boldsymbol{\alpha}(x))$, as described by Sensoy et al. (2018).

We were initially surprised to find that the Evidential NN has significantly worse KL divergence and calibration scores in Table 1 compared to other methods, but only a moderate reduction in classification error. After further investigation, we determined that this is because, in the presence of label noise between a few classes, the Evidential NN's regularization causes it to predict an almost-uniform distribution across all classes, even classes that never appear (as shown in Figure 24), due to predicting a Dirichlet distribution that is nearly uniform over the simplex. Additionally, because the denominator $\mathbf{1}^T\boldsymbol{\alpha}(x) + 1$ is always at least `num_classes` + 1, the variance estimate will never be larger than $\frac{0.5^2}{11} \approx 0.022$, and if the predicted probability is close to uniform, the variance estimate will be around $\frac{0.1 \cdot 0.9}{11} \approx 0.0082$. This may be smaller than the actual squared error of the predictor. (This occurs because the uniform distribution over a high-dimensional simplex actually has very low variance along each dimension.)

Cheat NN: We parameterize the output layer of the wide ResNet with $10 \times 10 = 100$ output classes. We then reshape them into a 10×10 matrix and add it to its transpose to enforce that it is symmetric, and then take a softmax over all rows and columns. To regularize this output and prevent negative variance estimates due to overfitting, we compute the eigenvalues of this probability matrix at each training step, then regularize any negative eigenvalues by multiplying their squared norm by 10.0. See Appendix E for additional discussion.

In contrast to the previous approaches, which average the loss over the two samples y_1, y_2 for each example seen, for our method we directly compute the log-likelihood of the *pair* of outputs, by indexing into the appropriate row and column of our joint probability matrix.

At test time, we compute $\hat{p}^\theta_{Y_1|X}(y|x)$ by marginalizing out one of the axes of our symmetric 10×10 matrix. We compute $\hat{V}^\theta_{\text{CHEAT}}(y|x)$ using

$$\hat{V}^\theta_{\text{CHEAT}}(y|x) = \hat{p}^\theta_{Y_1,Y_2|X}(y, y|x) - \hat{p}^\theta_{Y_1|X}(y|x)^2.$$

Note that, for hyperparameter tuning, we compute the KL divergence according to the predicted marginal distribution of Y_1

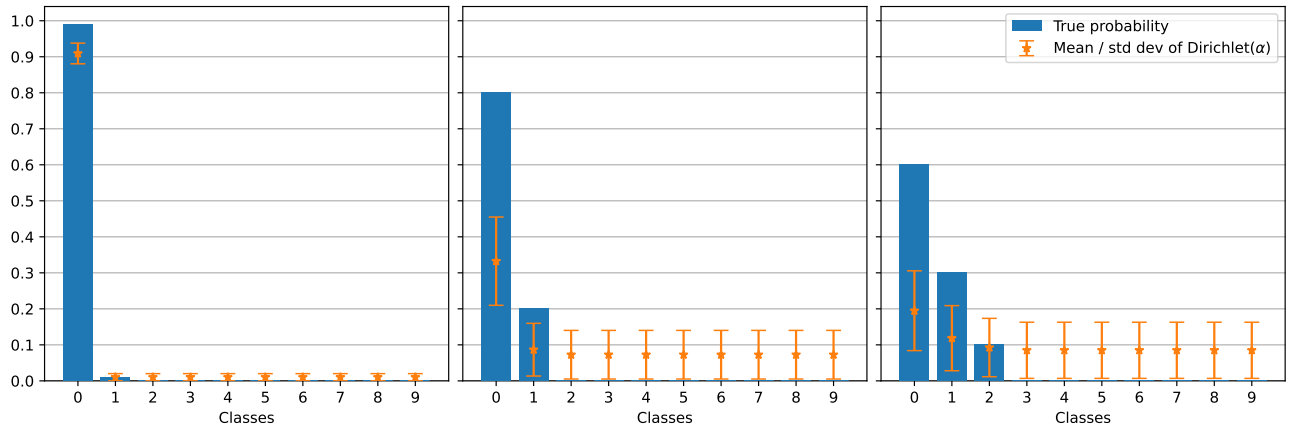

Figure 24. Optimizing the Evidential Deep Learning objective Sensoy et al. (2018) produces biased probability estimates in the presence of label noise. We let $\boldsymbol{\alpha} = 1 + \text{softplus}(\boldsymbol{v})$ and directly minimize the EDL objective with respect to $\boldsymbol{v}$, taking an expectation over a synthetic ground-truth label distribution (blue bars). We then visualize the mean and standard deviation of the learned distribution (orange). When the ground-truth distribution has aleatoric uncertainty, EDL both over-estimates the probability for never-observed classes, and under-estimates the distance from the true label distribution.

only, and compare it to the empirical distribution of all 50 annotator labels; we do not use the conditional $\hat{p}^{\theta}_{Y_2|Y_1,X}$ (or the joint $\hat{p}^{\theta}_{Y_1,Y_2|X}$) for hyperparameter tuning.

Cheat SNGP: We follow the same procedure as for Cheat NN, but use the SNGP variant of the wide ResNet architecture. This means we use the spectral normalization layers and random-feature Gaussian process output head. However, we do not compute any posterior covariance using the Gaussian process output head, and instead merely use the random features as a convenient parameterization for a deterministic output layer. This allows us to take advantage of the distance-awareness inductive biases in the SNGP architecture (Liu et al., 2020) without needing to approximate an actual posterior distribution.

F.2.3. Dataset variants

In addition to the original dataset, we consider three dataset variants: extra classes, scrambled, and extra classes + scrambled. Each of these variants is implemented by transforming either the images or the annotator labels for the tasks, and we apply the transformations to all of the dataset splits (pretraining, fine-tuning, validation, and test).

For the original and the extra classes + scrambled variants, we aggregate over eight independent training and evaluation runs. The average performance is shown in the main paper in Table 1, and the standard deviations are given in Table 3. For the other two variants, we only conduct a single training run; the results for these variants are given in Table 4.

We do not perform separate hyperparameter sweeps for each variant; instead we re-use the optimal hyperparameters for the unmodified dataset, and just run training phases 2, 4, and 5 (as described above). Example images from each of these dataset variants are shown in Figure 25.

Extra Classes: In this variant, we add label noise by artificially increasing the number of classes. For each of the classes in the original CIFAR-10 dataset, we create three new classes (e.g. dog $\rightarrow$ { dog-1, dog-2, dog-3}), and arbitrarily construct a distribution over them (e.g. $p(Y'|Y = \text{dog})$ for $Y' \in \{\text{dog-1, dog-2, dog-3}\}$) by sampling from Dirichlet([1, 1, 1]). This produces a classification problem with 30 classes instead of 10, where there is aleatoric variation between the sub-classes even for unambiguous images. The conditional distribution for each class is held fixed for all models and all dataset splits (i.e. it is treated as part of the data distribution itself).

When training with this variant, we increase the number of outputs of each method accordingly; methods that had output dimension 10 instead use output dimension 30, and our cheat-corrected models are configured to produce 30×30 joint matrices. We then sample a sub-class for each class in each training iteration, potentially using different sub-classes for the same image in different epoch. When evaluating metrics, we multiply the empirical distribution over the original labels and the closed-form conditional distribution for the sub-labels to construct a partially-empirical distribution over the sub-labels.

We note that this dataset has more aleatoric variation, but it shouldn't require more capacity, since the noise is added in an

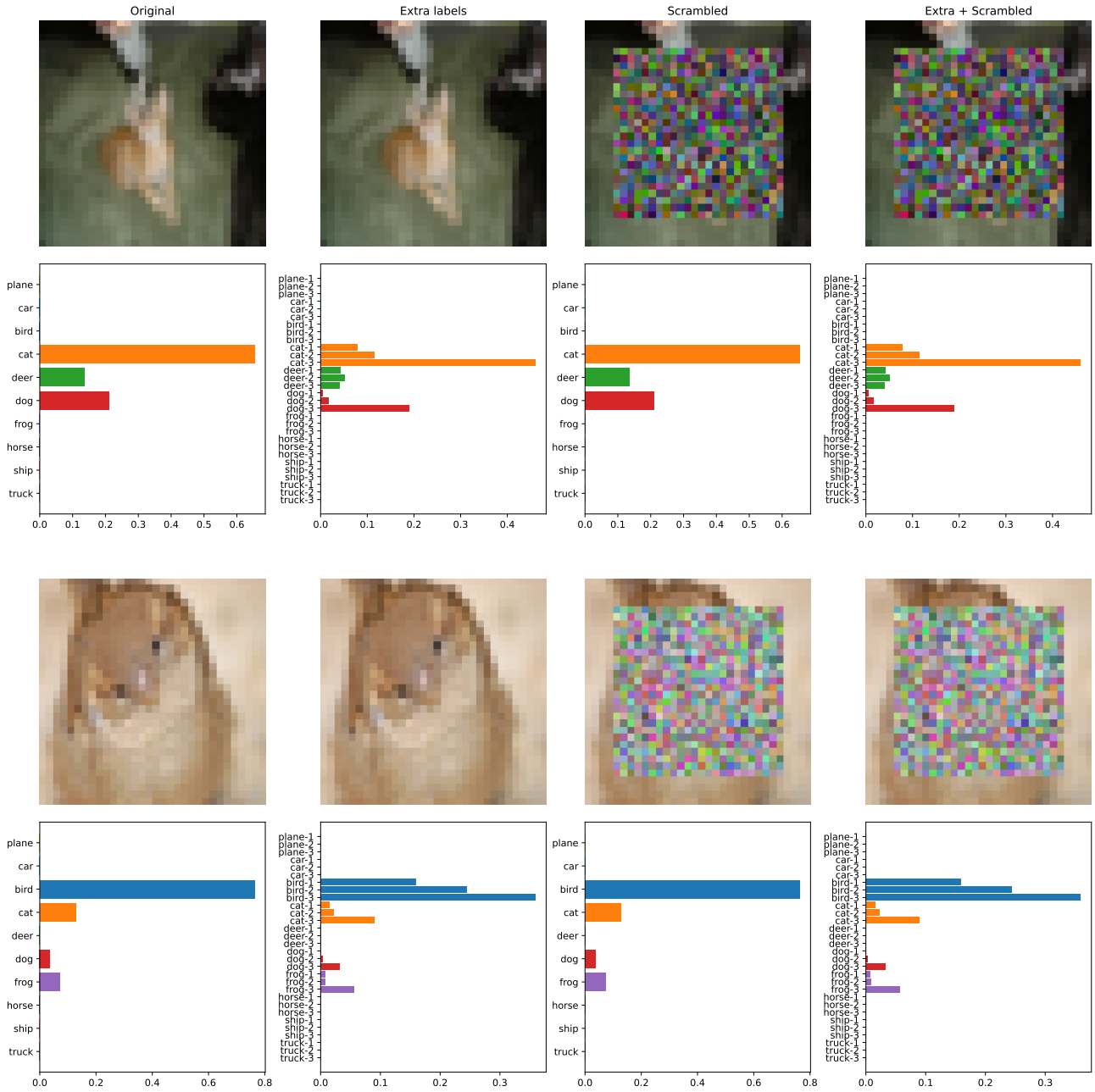

Figure 25. Input images X and ground-truth annotator label distributions $p(Y|X)$ for two images in the CIFAR-10H dataset, selected due to having natural aleatoric uncertainty in the label distribution. The "Extra classes" variant modifies the label distribution, whereas the "Scrambled" variant scrambles the center patch of the image.

Table 3. Full results from Table 1 with mean and standard deviation across eight training runs. All metrics summed across classes except Acc and KL.

	CIFAR-10H					
METHOD	**ECE-2**	$\mathbb{E}[\hat{v}^\theta]$	$\mathbb{E}[(\hat{p}^\theta - p)^2]$	ECE-1	ACC	KL
NAIVE NN	0.076 ± 0.003	0.142 ± 0.001	0.065 ± 0.002	0.017 ± 0.003	93.94 ± 0.17	0.179 ± 0.003
NN ENSEMBLE	0.039 ± 0.000	0.014 ± 0.000	0.053 ± 0.000	0.029 ± 0.002	94.90 ± 0.09	0.152 ± 0.001
EVIDENTIAL DL	0.377 ± 0.001	0.053 ± 0.000	0.430 ± 0.002	1.038 ± 0.004	88.45 ± 0.23	1.087 ± 0.003
SNGP COV.	0.048 ± 0.001	0.005 ± 0.000	0.052 ± 0.001	0.020 ± 0.001	94.89 ± 0.15	0.153 ± 0.001
EPINET	0.056 ± 0.002	0.015 ± 0.002	0.071 ± 0.001	0.020 ± 0.002	93.40 ± 0.26	0.189 ± 0.002
CHEAT NN	0.018 ± 0.001	0.052 ± 0.000	0.068 ± 0.001	0.029 ± 0.003	93.60 ± 0.18	0.182 ± 0.002
CHEAT SNGP	0.009 ± 0.002	0.054 ± 0.001	0.052 ± 0.001	0.022 ± 0.002	94.90 ± 0.14	0.149 ± 0.001

	W/ EXTRA CLASSES, SCRAMBLED				
METHOD	**ECE-2**	$\mathbb{E}[\hat{v}^\theta]$	$\mathbb{E}[(\hat{p}^\theta - p)^2]$	ECE-1	KL
NAIVE NN	0.521 ± 0.002	0.682 ± 0.003	0.161 ± 0.002	0.068 ± 0.004	0.706 ± 0.009
NN ENSEMBLE	0.134 ± 0.001	0.014 ± 0.000	0.148 ± 0.001	0.032 ± 0.001	0.647 ± 0.003
EVIDENTIAL DL	0.387 ± 0.000	0.031 ± 0.000	0.418 ± 0.000	0.794 ± 0.009	2.356 ± 0.000
SNGP COV.	0.112 ± 0.003	0.033 ± 0.003	0.145 ± 0.001	0.057 ± 0.003	0.634 ± 0.006
EPINET	0.089 ± 0.009	0.087 ± 0.009	0.163 ± 0.003	0.075 ± 0.003	0.712 ± 0.010
CHEAT NN	0.022 ± 0.001	0.134 ± 0.001	0.154 ± 0.001	0.072 ± 0.005	0.672 ± 0.005
CHEAT SNGP	0.011 ± 0.001	0.153 ± 0.002	0.150 ± 0.001	0.044 ± 0.003	0.650 ± 0.006

image-agnostic way.

Scrambled: In this variant, we increase the task difficulty by applying a fixed permutation to the pixels in the center of the image. This permutation applies across all channels and all pixels except for a 4-pixel border around the sides of the image. Since the content is usually in the center of the image, and since a ResNet is a convolutional architecture, this permutation is likely to interfere with the inductive biases of all of the methods, and may cause them to focus on less-informative features in the image border.

Since our permutation is invertible, it is always possible in principle to reconstruct the original image from the scrambled form. This means that the true conditional $p(Y|X)$ is not affected by using a scrambled view of X, so all of the additional uncertainty from this transformation is epistemic in nature.

We apply this permutation after the AugMix augmentations during training.

Extra + Scrambled: We apply both of the transformations above together.

F.2.4. EVALUATION METRICS

ECE-2: Our primary evaluation metric is the expected second-order calibration error between the predicted epistemic variance and the actual squared difference between the predicted probability and $p(Y|X)$. In other words, we wish to evaluate how closely the following correspondence holds:

$$\mathbb{E}\Big[\big(p(y|X) - \hat{p}^\theta_{Y|X}(y|X)\big)^2 \;\Big|\; \hat{V}^\theta(y|X)\Big] \stackrel{?}{\approx} \hat{V}^\theta(y|X).$$

(We expect this correspondence to hold because it is a special case of Theorem 4.2 where the event A is $\hat{V}^\theta(y|X) = c$ for each possible $c \in \mathbb{R}$.) Specifically, we focus on expected calibration error, which has the form

$$\mathbb{E}\left[\left|\mathbb{E}\Big[\big(p(y|X) - \hat{p}^\theta_{Y|X}(y|X)\big)^2 \;\Big|\; \hat{V}^\theta(y|X)\Big] - \hat{V}^\theta(y|X)\right|\right]$$

We estimate this using expected calibration error over 100 equal-probability bins, based on the quantiles of the predicted probability $\hat{V}^\theta(y|X)$, which we compute as follows:

Table 4. Results for the additional CIFAR-10H dataset variants we consider, which include only one of the two increased difficulty types each. Results within 2x of best ECE-2 in bold. All metrics summed across classes except Acc and KL.

	Extra Classes					Scrambled				
Method	**ECE-2**	$\mathbb{E}[\hat{v}^\theta]$	$\mathbb{E}[(\hat{p}^\theta - p)^2]$	ECE-1	KL	**ECE-2**	$\mathbb{E}[\hat{v}^\theta]$	$\mathbb{E}[(\hat{p}^\theta - p)^2]$	ECE-1	KL
Naive NN	0.540	0.574	0.034	0.02	0.18	**0.051**	0.354	0.314	0.07	0.69
NN Ensemble	0.020	0.007	0.027	0.03	0.15	0.261	0.028	0.289	0.04	0.64
Evidential DL	0.386	0.031	0.417	1.14	2.34	0.561	0.068	0.629	0.97	1.59
SNGP Cov.	**0.017**	0.030	0.026	0.03	0.15	0.271	0.014	0.285	0.06	0.63
Epinet	0.054	0.083	0.041	0.04	0.20	0.271	0.044	0.314	0.08	0.69
Cheat NN	**0.010**	0.029	0.037	0.04	0.19	**0.056**	0.252	0.303	0.08	0.66
Cheat SNGP	**0.009**	0.032	0.028	0.02	0.15	**0.029**	0.280	0.286	0.05	0.62

1. For each example x, for each class y, compute the variance estimate $\hat{V}^\theta(y|x)$, and an unbiased estimate of the squared error $p(y|X) - \hat{p}^\theta_{Y|X}(y|X)\big)^2$ using the K annotations for this image x:

$$\begin{aligned}\textsc{SqErrEst}(y, \hat{p}^\theta_{Y|X}(y|x), [y_i]_{i=1}^K) &= \hat{p}^\theta_{Y|X}(y|x)^2 - 2\hat{p}^\theta_{Y|X}(y|x)\frac{|\{i : y_i = y\}|}{K} + \frac{|\{(i,j) : i \neq j, y_i = y, y_j = y\}|}{K(K-1)} \\ &\approx \hat{p}^\theta_{Y|X}(y|x)^2 - \hat{p}^\theta_{Y|X}(y|x)p(y|x) + p(y|x)^2 = (\hat{p}^\theta_{Y|X}(y|x)^2 - p(y|x))^2\end{aligned}$$

   K is the number of annotator labels for this image, which is always at least 50 but is sometimes greater. When evaluating for the "extra classes" variant, we compute the modified estimate

$$\begin{aligned}\textsc{SqErrEst}(y', \hat{p}^\theta_{Y|X}(y'|x), [y_i]_{i=1}^K) = \hat{p}^\theta_{Y|X}(y'|x)^2 &- 2\hat{p}^\theta_{Y|X}(y'|x)p(y'|y)\frac{|\{i : y_i = y\}|}{K} \\ &+ p(y'|y)^2\frac{|\{(i,j) : i \neq j, y_i = y, y_j = y\}|}{K(K-1)}\end{aligned}$$

   where y' is one of the three sub-classes corresponding to the original class y.

2. Sort all of the (x, y) pairs in ascending order of $\hat{V}^\theta(y|x)$. Note that each x appears multiple times in this ordering, due to the different possible labels y.

3. Divide the examples into 100 evenly-sized bins, each of which correspond to an empirical quantile range of 1%.

4. Compute the average $\overline{v}_{B_i}$ of $\hat{V}^\theta(y|x)$ for all examples (x, y) in each bin B_i. Also compute the average $\overline{\textsc{SqErrEst}}_{B_i}$ of the error estimates $\textsc{SqErrEst}(y, \hat{p}^\theta_{Y|X}(y|x), [y_i]_{i=1}^K)$ for those same examples.

5. Let

$$T = \frac{1}{N}\sum_{B_i} |B_i| \cdot \left|\overline{v}_{B_i} - \overline{\textsc{SqErrEst}}_{B_i}\right|,$$

   where N is the total number of test set examples and $|B_i|$ is the size of the ith bin (approximately N/100).

6. Return ECE-V $= C \cdot T$ where C is the number of classes.

We scale up the expected calibration error by the number of classes in the last step so that our final metric represents a sum over classes instead of an average over classes, since we usually care about the full vector of predictions rather than the prediction for a single random class. Note that some papers evaluate expected calibration error for the most likely predicted class only, which avoids this problem (Perez-Lebel et al., 2022). However, it is not obvious that this criterion makes sense when evaluating epistemic uncertainty, especially in the presence of significant aleatoric uncertainty. In principle, we could also compute a separate calibration score for each class and then add them together at the end, instead of averaging over them and then scaling the average, but we choose not to do this because of the small size of our dataset (which would potentially make per-class calibration error estimates very noisy).

We also note that, in general, the binning procedure will under-estimate the exact expected calibration error, although this can be avoided by using the binned outputs instead of the original ones (Kumar et al., 2019).

$\mathbb{E}[\hat{v}^\theta]$, $\mathbb{E}[(\hat{p}^\theta - p)^2]$: To compute these values, we use the same estimates from the ECE-V computation, but instead of averaging differences over each bin, we simply compute separate sums of $\hat{V}^\theta(y|x)$ and $\textsc{SqErrEst}(y, \hat{p}^\theta_{Y|X}(y|x), [y_i]_{i=1}^K)$. We divide by the total number of examples, so that these numbers again reflect sums over all classes (30 classes in this case).

ECE-1: We also estimate the expected calibration error for the ordinary predictions, again using 100 quantile bins, combining classes together, and scaling up by the number of classes. We apply the same procedure as for ECE-2, except that we use the predicted probability estimates $\hat{p}^\theta_{Y|X}(y|x)$ instead of the variance estimates $\hat{V}^\theta(y|x)$, and we use the empirical probability $\frac{|\{i:y_i=y\}|}{K}$ instead of the squared error measurement $\textsc{SqErrEst}(y, \hat{p}^\theta_{Y|X}(y|x), [y_i]_{i=1}^K)$.

KL: Finally, we compute the average KL divergence between the empirical probability distribution of the annotator labels and the model predictions

$$D_{KL}\Big(p_{\mathcal{D}}(y|x) \,\Big\|\, \hat{p}^\theta_{Y|X}(y|x) \Big) = \frac{|\{i : y_i = y\}|}{K} \left(\log \hat{p}^\theta_{Y|X}(y|x) - \log \frac{|\{i : y_i = y\}|}{K} \right),$$

where $p_{\mathcal{D}}(y|x)$ is the distribution of annotator labels for image x, e.g.

$$p_{\mathcal{D}}(y|x) = \frac{|\{i : y_i = y\}|}{K}$$

where $[y_i]_{i=1}^K$ is the collection of annotator labels for image x.

We use this KL divergence metric to tune the hyperparameters of each method.

F.3. Digits of π

F.3.1. TRAINING DATA

To construct the queries X, we sample digit offsets I according to a mixture of geometric random variables: we sample

$$Q \sim \text{Uniform}(0.001, 0.1), \qquad I \sim \text{Geometric}(Q),$$

then keep I if it is less than 10,000 and reject it otherwise. We then embed this as a tokenized sequence of the form "Tell me about digit 0 0 1 4 of pi.", where I is zero-padded to four digits long.

Given I, we then look up the actual Ith digit after the decimal point in π (so digit 1 is 1, digit 2 is 4, digit 3 is 1, digit 4 is 5, etc.).

Depending on the value d of this digit, we then sample responses Y from the following hand-written probabilistic context-free grammar (with mostly randomly-chosen weights):

STATEMENT(d)	→ INTRO VALUE(d)	with probability 0.99
	→ "Reply hazy, try again"	with probability 0.01
INTRO	→ "It's"	with probability 0.138
	→ "It is"	with probability 0.086
	→ "That's"	with probability 0.218
	→ "That is"	with probability 0.185
	→ "Sure, it's"	with probability 0.096
	→ "Sure, it is"	with probability 0.02
	→ "Sure, that's"	with probability 0.17
	→ "Sure, that is"	with probability 0.087
VALUE(d)	→ SAY-DIGIT(d)	with probability 0.56
	→ "an" EVEN-ODD(d) "number"	with probability 0.19
	→ "spelled" SPELL(d)	with probability 0.13
	→ "spelled with" SPELL-LENGTH(d) "letters"	with probability 0.9
SAY-DIGIT(d)	→ DIGIT(d)	with probability 0.616
	→ "the number" DIGIT(d)	with probability 0.384
DIGIT(d)	→ DIGIT-NAME(d)	with probability 0.323
	→ DIGIT-VAL(d)	with probability 0.677

The nonterminals DIGIT-NAME(d), DIGIT-VAL(d), EVEN-ODD(d), SPELL(d), and SPELL-LENGTH(d) depend on the digit:

- DIGIT-NAME takes values "zero", "one", "two", ...
- DIGIT-VAL takes values "0", "1", "2", ...
- EVEN-ODD takes values "even", "odd", "even", ...
- SPELL takes values "Z E R O", "O N E", "T W O", ...

- SPELL-LENGTH takes values "4" (the number of letters in "zero"), "3" (the number of letters in "one"), "3" (the number of letters in "two"), ...

Using this context free grammar, we can both sample sequences and look up the true probability of any given sequence. In particular, given a statement, we can look up whether it is in the support of the grammar given the true digit value.

We also maintain a lookup table of semantically-equivalent sentences. When doing this, we treat as equivalent any two sentences that differ only based on how they expanded INTRO, SAYDIGIT, and DIGIT. So, for instance, "Sure, it's the number 3" and "That is three" are equivalent, and "It's spelled T H R E E" and "Sure, it's spelled T H R E E" are equivalent, but "Sure, it's the number three" and "It's spelled T H R E E" are *not* judged equivalent (one is about the value and the other is about the spelling). Similarly "It's spelled F O U R" and "It's spelled with 4 letters" are not equivalent.

We train our model by concatenating X and two samples Y_1, Y_2. Before concatenation, we pad Y_1 and Y_2 out to a constant length, producing examples of the form

```
<BOS> Tell me about digit 0 1 2 6 of pi. <SEP>
That's the number four _ _ _ _ _ _ That's spelled with 4 letters _ _ _ _ _
```

```
<BOS> Tell me about digit 0 0 4 8 of pi. <SEP>
Sure, that is an odd number _ _ _ _ It's 5 _ _ _ _ _ _ _ _
```

```
<BOS> Tell me about digit 0 0 1 2 of pi. <SEP>
Sure, that's 9 _ _ _ _ _ _ _ It is the number nine _ _ _ _ _
```

```
<BOS> Tell me about digit 0 0 1 5 of pi. <SEP>
Sure, that is 3 _ _ _ _ _ _ That is spelled T H R E E _ _
```

We use a tabular vocabulary, where each word or letter that could possibly be generated by the above process has its own token index.

F.3.2. Model architecture and training details

Our model architecture is a 6-layer causally-masked pre-LayerNorm Transformer (Vaswani et al., 2017; Xiong et al., 2020), with 8 attention heads, an embedding dimension of 512, an MLP dimension of 2048, a per-head embedding dimension of 64, and fixed sinusoidal positional embeddings.

We train this model for 50,000 training iterations at batch size 1024 using the AdamW optimizer (Loshchilov & Hutter, 2017) with 1,000 warmup steps, a maximum learning rate of 2×10^{-5}, and a cosine decay schedule. We use the ordinary maximum-likelihood objective, and apply masking so that only the tokens after <SEP> are scored (so the model does not have to learn to predict X). Our implementation is in JAX (Bradbury et al., 2018) and uses Optax for optimization (DeepMind et al., 2020).

F.3.3. Sampling and filtering

After training our model, we iterate through all of the digit offsets from 1 to 3000, and sample 120 statements from the model's approximate conditional $\hat{p}^{\theta}_{Y_1|X}(\cdot|x)$. We sample at temperature 1 but mask out any tokens with predicted probability less than 0.005 during sampling. We note that the model has learned to sample pairs (Y_1, Y_2), but we interrupt generation after it has generated Y_1; thus each of the 120 statements are drawn independently and identically distributed from $\hat{p}^{\theta}_{Y_1|X}(\cdot|x)$, not $\hat{p}^{\theta}_{Y_2|Y_1,X}$).

We first check whether each sample was correct, where we judge a sample as correct if it had a nonzero probability under $p(Y|X)$ (the context-free grammar described in Appendix F.3.1). We then assign scores to each sample:

- For the total probability ranking, we rank by the probability $\hat{p}^{\theta}_{Y_1|X}(y_1|x)$ of the sample under the model.

- For average token log probability, we divide $\log \hat{p}^{\theta}_{Y_1|X}(y_1|x)$ by the length $|y_1|$ of the sample; this is thus an average of the log probabilities of each token, and approximates a "rate" of log probability (Malinin & Gales, 2020).

- When clustering into groups of 10, we split the 120 samples for each digit into 12 groups of size 10 each, then assign a score to each sample based on the number of other samples in the same group that were semantically equivalent (according to the criterion in Appendix F.3.1). So, if 4 out of the first 10 samples were semantically equivalent, each of those samples would get a score of 4. (Malformed samples that could not possibly appear under the data distribution are given a score 1.)

- When clustering into groups of 120, we assign a score to each sample based on the number of other samples among all 120 that were semantically equivalent.

- When ranking based on our epistemic confidence measure $C^{\theta}_{\text{CHEAT}}$, we evaluate $\log \hat{p}^{\theta}_{Y_1|X}(y|x)$ and $\log \hat{p}^{\theta}_{Y_2|Y_1,X}(y|y,x)$ by concatenating the padded output y with itself when scoring using the model, and separately summing the log-probabilities for Y_1 and Y_2. We then exponentiate the difference of these probabilities to evaluate $C^{\theta}_{\text{CHEAT}}$, and finally assign a score of $-|1 - C^{\theta}_{\text{CHEAT}}|$ so that examples closer to 1 have a higher score.

We additionally explored a number of alternative ranking strategies for the case where $C^{\theta}_{\text{CHEAT}} > 1$. According to Proposition 4.4, $C^{\theta}_{\text{CHEAT}}$ will never be greater than 1 if $\hat{p}^{\theta}_{Y_1,Y_2|X}$ is calibrated, so our theoretical results do not provide any particular guidance for how to rank these samples. We plot four options in Figure 26: $1 - C^{\theta}_{\text{CHEAT}}$ (the simplest, but nonsensical when $C^{\theta}_{\text{CHEAT}}$ is very large), $|1 - C^{\theta}_{\text{CHEAT}}|$ (used in the main paper results), $1 - \min\{1, C^{\theta}_{\text{CHEAT}}\}$, and $1 - \min\left\{C^{\theta}_{\text{CHEAT}}, \frac{1}{C^{\theta}_{\text{CHEAT}}}\right\}$. Using $1 - C^{\theta}_{\text{CHEAT}}$ alone leads to high hallucination rates when using very strict thresholds, because the only samples that are kept are the outliers.

F.3.4. Investigating model samples

For each score type, we sort the samples (randomizing in the case of ties), and then compute the running hallucination rate (fraction of samples seen so far that actually had $p_{Y|X}(y|x) = 0$) and response rate (total fraction of samples seen so far) over prefixes of the sorted ordering; the results are shown in Figure 5 (right) in the main paper.

To get a better understanding of the relationship between the model's behavior, its accuracy, and its cheat-corrected epistemic confidence, we conduct a deeper study of the scores and accuracies assigned to each of the digits.

We start by visualizing some samples drawn from the model directly, colored based on the log-probability of each token. In Figures 7 and 8 (in Appendix A), we show samples of Y_1 and Y_2 generated by the model. Note that the samples of Y_2 are not actually used under our uncertainty-quantification scheme. However, visualizing these samples reveals an interesting behavior: when querying digits that the model does not know, the samples Y_1 often show "hallucinations" of plausible facts, but the corresponding Y_2 is almost always *consistent* with Y_1; the two samples do not contradict one another. This means that the model has learned to "cheat" well; it is able to condition on the information in Y_1 to make a more consistent prediction of Y_2. There are, however, a few exceptions where Y_1 and Y_2 are inconsistent or incoherent; in this case we find that Y_2 tends to be more correct than Y_1.

In Figures 9 and 10 (in Appendix A), we next visualize the log-probabilities of each token when reuse the sampled Y_1 sequences as Y_2, by concatenating each Y_1 with itself; this is how we compute our confidence scores $C^{\theta}_{\text{CHEAT}}(y|x)$. We see that conditioning on Y_1 does not usually significantly alter the probability of tokens with aleatoric variation (e.g. the initial stylistic tokens), but usually raises the probability of tokens with epistemic prediction error (e.g. tokens that state facts about the digit). This means the difference between the two log probabilities is usually a good indicator of the unknown parts of the original samples.

There are a few samples which show the opposite pattern, where the likelihood *decreases* in the second sample (in the last row of each figure). This seems to occur when the originally sampled Y_1 was incorrect and had a very low probability, whereas the prediction for Y_2 was more accurate. This pattern of an incorrect Y_1 but correct Y_2 is an indication of *miscalibration* with respect to the paired outcomes (Y_1, Y_2): if the model was truly conditioning on some property $\Phi(X)$ of the input query, its predictions on Y_1 should be just as accurate as its prediction of Y_2, and it should never predict an inconsistent Y_1, Y_2 pair that cannot occur under the data distribution. These samples tend to interfere with our epistemic confidence metric, and result in predicted confidences greater than 1 (and negative predicted variances).

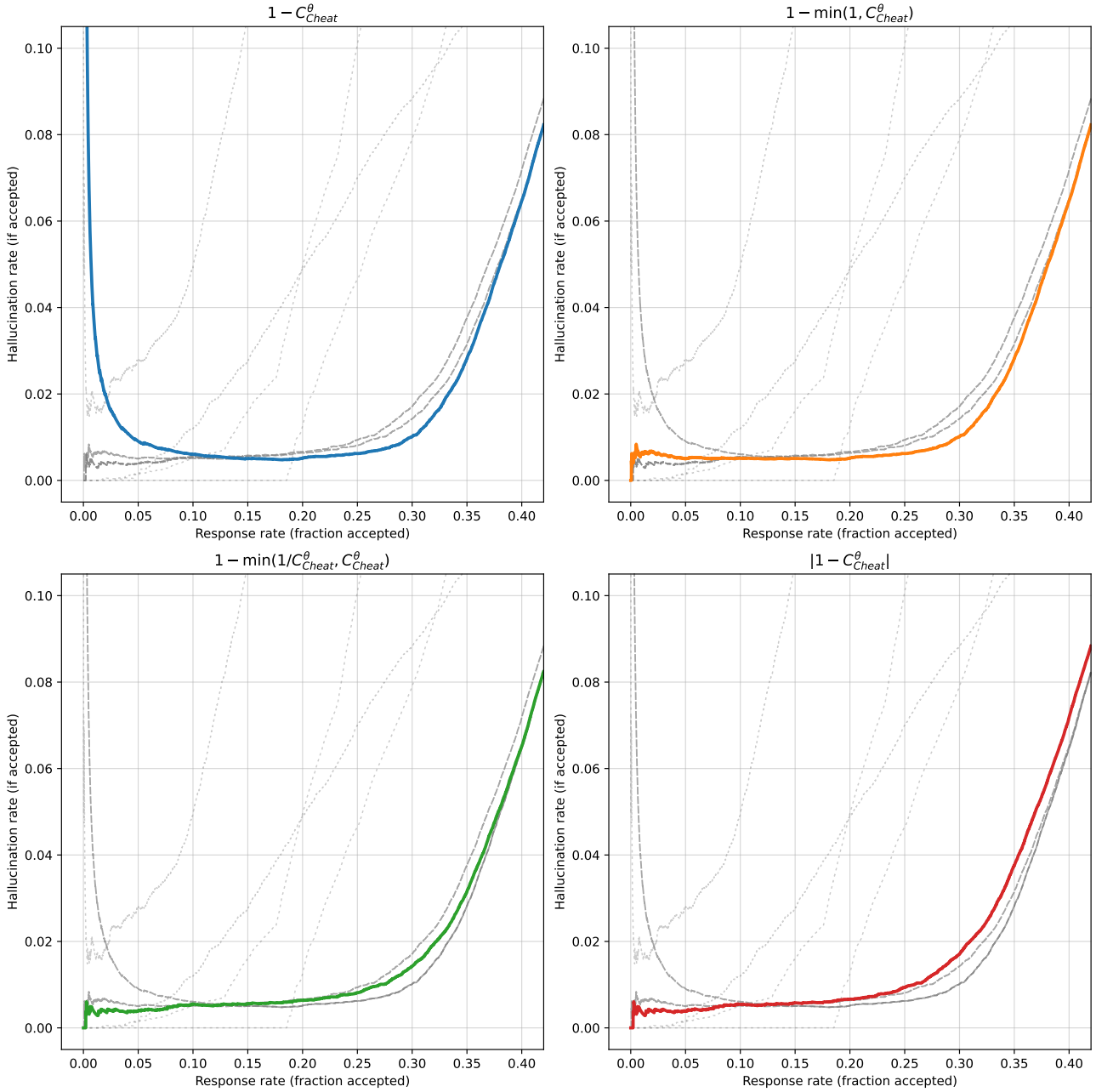

Figure 26. Hallucination rates when ranking by alternative versions of $C^{\theta}_{\text{CHEAT}}$, each of which agree with $C^{\theta}_{\text{CHEAT}}$ when it is less than 1 but handle $C^{\theta}_{\text{CHEAT}} > 1$ differently. For comparison, the original baselines from Figure 5 are shown as dotted lines.

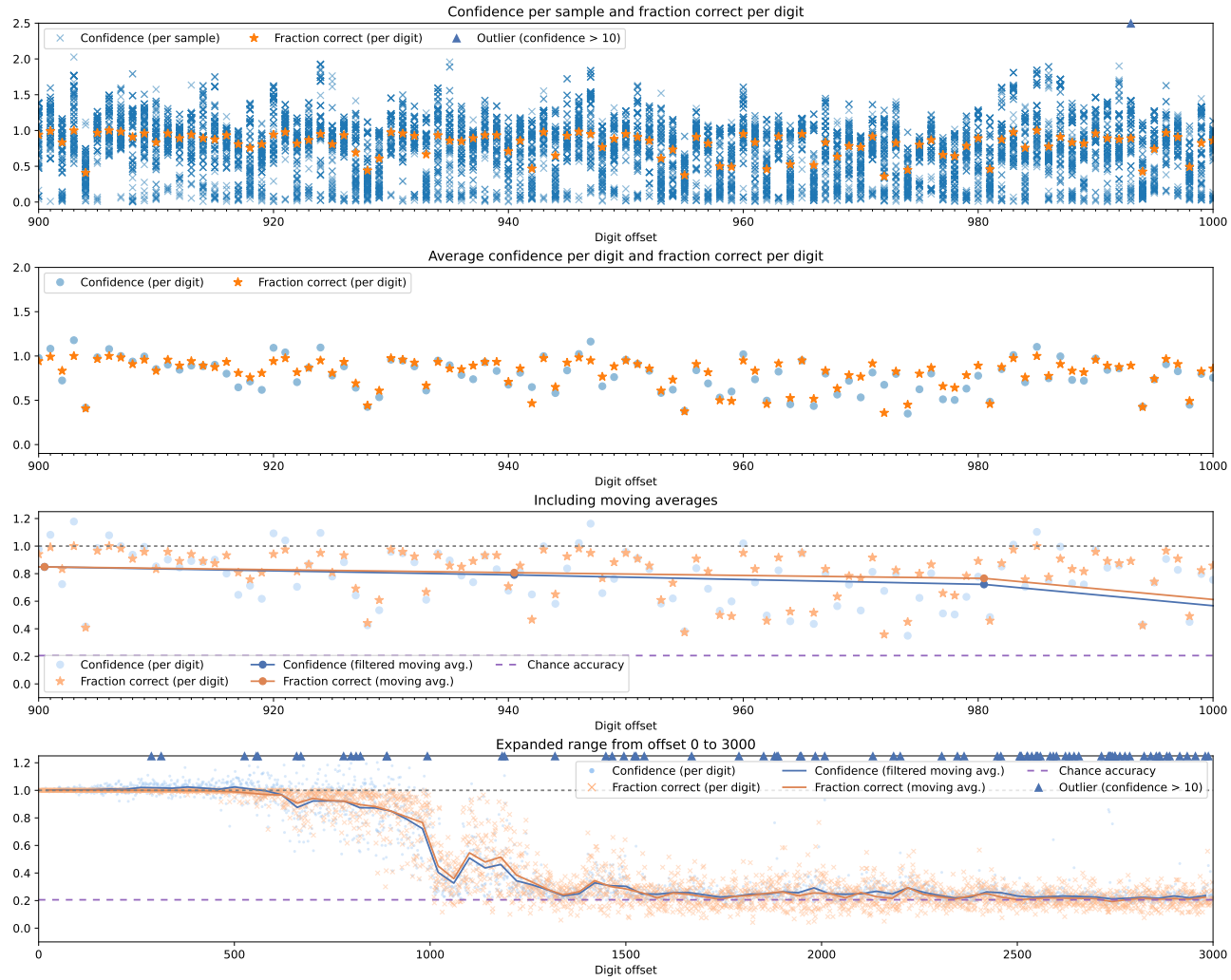

Figure 27. Predicted epistemic confidence closely tracks the model's actual accuracy after removing outliers. Row 1: We compute the confidence for each of the 120 samples for each digit, and also compute the fraction of those 120 samples that are correct. Row 2: We take the average confidence over all of the samples for each digit, ignoring outliers. Row 3: We divide digit offsets into groups of 40 digits, and compute the average of both confidence and fraction correct. Row 4: Zoomed out version of row 3, showing the full sequence.

F.3.5. Relationship between confidence and correctness

We also investigate the relationship between the epistemic confidence and the correctness of generated samples. This is somewhat complex, because each individual sample is either correct or not correct, and the model may assign different confidences to each sample. Determining whether the model's confidence was appropriate thus requires some sort of aggregation.

Figure 28 shows an expanded version of the confidence-vs-hallucination-rate plot in Figure 5 (left). Overall, when the confidence is less than 1, the quantity $1 - C^{\theta}_{\text{CHEAT}}$ is a good estimate of hallucination rate. Samples with confidence close to 1 tend to be correct, but samples with extremely large confidence values tend to be wrong. We find that most of these outliers are due to having an incorrect Y_1 but fixing the mistake in Y_2, as detailed in the previous section.

To show how confidence varies based on digit offset, we additionally aggregate these across nearby digits, since we know from the data distribution that nearby digits appear with similar frequency and should thus be similarly difficult. We compute the fraction of samples that are correct for each digit, and compare it to the average confidence of the samples. To get a meaningful aggregate confidence measurement, we need to throw out samples with extremely large confidences due to miscalibration. The results are shown in Figure 27.

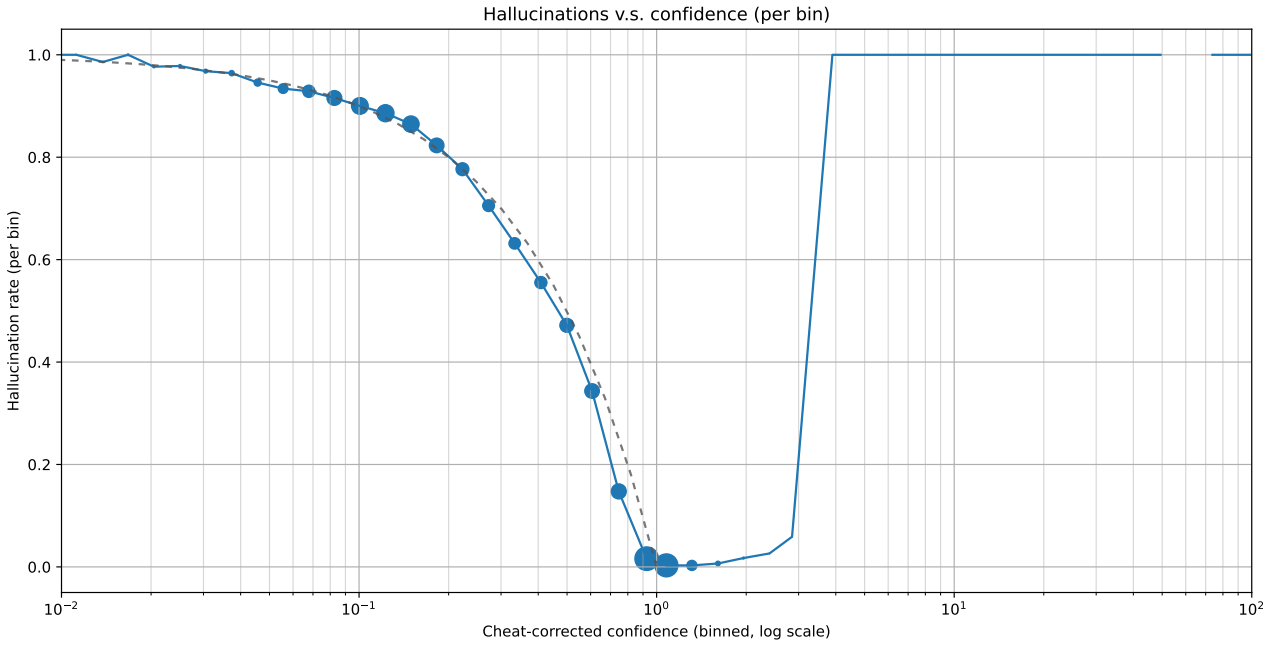

Figure 28. Expanded version of Figure 5 (left), which shows the hallucination rate of the samples in different confidence bins, with dot size proportional to the number of samples with that confidence. We use a logarithmic scale for $C^{\theta}_{\text{CHEAT}}$ to include the outliers which are much larger than 1; the prediction of Theorem 4.5 is shown as a dashed line. We see that confidences between 1 and 2 usually indicate a correct answer, but this quickly falls off, and most samples with extremely large $C^{\theta}_{\text{CHEAT}}$ values are usually incorrect.

Figure 29. Expert policies for the frozen lake task, depending on the location of the unsafe patch. Arrow length is proportional to probability of taking that action in each state.

F.4. Offline RL - Frozen Lake

F.4.1. Environment and Expert Policies

We represent the Frozen Lake environment (Figure 6) as a graph, where each gridworld cell is a node in the graph, and there are four outgoing connections from each node to the adjacent nodes in each direction. Costs of each transition are determined by the destination state:

- Moving onto the goal state (green square) gives a reward of 40 and ends the episode.
- Moving onto one of the non-lake squares (white border) gives a reward of -3.
- Moving onto one of the eight non-central lake squares gives a reward of -5.
- Moving onto the center lake square gives a reward -10.

These costs were chosen so that cutting across the center of the lake gives slightly more reward than going around it ($-5 - 10 - 5 + 40 = 20$ vs $-3 \times 7 + 40 = 19$), but only slightly.

For each of the nine possible locations of the unsafe patch, we then solve for the optimal entropy-regularized tabular policy by iterating the soft Q-learning Bellman backup operator (Schulman et al., 2017) with a discount rate of 0.9 and a temperature of 2.5. We explicitly disallow moving onto the unsafe patch or leaving the bounds of the gridworld by assigning $-\infty$ reward to each; the resulting expert policy never takes those actions.

The resulting expert policies for each of the 9 possible unsafe patch locations are shown in Figure 29.

F.4.2. Imitation Learning

Our model architecture for this task is a 12-layer causally-masked pre-LayerNorm Transformer (Vaswani et al., 2017; Xiong et al., 2020) based on GPT-2 (Radford et al., 2019), with 12 attention heads, an embedding dimension of 768, an MLP dimension of 3072, a per-head embedding dimension of 64, and fixed sinusoidal positional embeddings.

For each example, we first tokenize the model's view of the environment, using a single token per gridworld cell. In 50% of the examples, we mark the unsafe region with a token "C" identifying it, and the safe parts of the lake with "I". In the other

50%, we mark all potentially-unsafe regions with a "?". We also include tokens for the start state ("S"), goal state ("G") and border states ("P"). (Since these tokens are constant across all examples, they are likely not important to include, but we include them for simplicity.)

Loosely inspired by the setup of Chen et al. (2021), we next represent the expert trajectories as sequences of states and actions. We do not tokenize the rewards, since our objective is simply to imitate the expert trajectories. We also concatenate *two* independent samples together, padding them to a maximum length of 16 steps each.

This produces examples of the following form (with each word mapped to its own token index, and padding denoted by "_"):

```
P P P P P
P I I I P
S I I C G
P I I I P
P P P P P
<SEP> c0 r2 down c0 r3 up c0 r2 down c0 r3 up c0 r2 down c0 r3 down c0 r4 right
c1 r4 up c1 r3 right c2 r3 right c3 r3 right c4 r3 up FINISH _ _ _ _ _ _ _ _ _ _
_ _ _
<SEP> c0 r2 right c1 r2 right c2 r2 down c2 r3 up c2 r2 down c2 r3 left c1 r3
down c1 r4 right c2 r4 right c3 r4 right c4 r4 left c3 r4 right c4 r4 up c4 r3
up FINISH _ _ _ _ _ _ _
```

```
P P P P P
P ? ? ? P
S ? ? ? G
P ? ? ? P
P P P P P
<SEP> c0 r2 right c1 r2 right c2 r2 right c3 r2 right FINISH _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
<SEP> c0 r2 up c0 r1 right c1 r1 down c1 r2 up c1 r1 down c1 r2 up c1 r1 up c1
r0 down c1 r1 down c1 r2 right c2 r2 right c3 r2 right FINISH _ _ _ _ _ _ _ _ _
_ _ _ _
```

We train by maximizing log-likelihood under a standard autoregressive training setup. We only train it to imitate the sequences of states and actions, by masking out all tokens prior to the <SEP> token. We train this model for 50,000 training iterations at batch size 512 using the AdamW optimizer (Loshchilov & Hutter, 2017) with 1,000 warmup steps, a maximum learning rate of 2×10^{-5}, and a cosine decay schedule. Our implementation is in JAX (Bradbury et al., 2018) and uses Optax for optimization (DeepMind et al., 2020).

We then sample trajectories from the model at temperature 0.9, conditioning on either a full or partial view of the environment, and compute the cheat-corrected epistemic confidence for each. For our "cheat-corrected rejection sampling" decoding strategy, we reject any sample with $|1 - C^{\theta}_{\text{CHEAT}}| > 0.05$ and resample it; rejected samples are shown as dashed lines in Figure 6 in the main paper. For our "cheat-corrected top-1 search" decoding strategy, we sample 6400 samples, then identify the sample y with the largest predicted probability $\hat{p}^{\theta}_{Y_1|X}(y|x)$ subject to the constraint $|1 - C^{\theta}_{\text{CHEAT}}| \leq 0.05$.

We show additional trajectories sampled by our model, along with their confidences, in Figures 11 to 13 (in Appendix A).