End-to-End Conformal Calibration for Robust Grid-Scale Battery Storage Optimization

Christopher Yeh[*], **Nicolas Christianson**[*], **Adam Wierman, Yisong Yue**
Department of Computing and Mathematical Sciences
California Institute of Technology
Pasadena, CA 91125
`{cyeh,nchristi,adamw,yyue}@caltech.edu`

Abstract

The rapid proliferation of intermittent renewable electricity generation demands a corresponding growth in grid-scale energy storage systems to enable grid decarbonization. To encourage investment in energy storage infrastructure, storage operators rely on forecasts of electricity prices along with uncertainty estimates to maximize profit while managing risk. However, well-calibrated uncertainty estimates can be difficult to obtain in high-capacity prediction models such as deep neural networks. Moreover, in high-dimensional settings, there may be many valid uncertainty estimates with varied performance profiles—*i.e.*, not all uncertainty is equally valuable for downstream decision-making. To address this challenge, this paper develops an end-to-end framework for conditional robust optimization, with robustness and calibration guarantees provided by conformal prediction. We represent arbitrary convex uncertainty sets with sublevel sets of partially input-convex neural networks, which are learned as part of our framework. We demonstrate the value of our approach for robust decision-making on a battery storage arbitrage application.

1 Introduction

Renewable wind and solar electricity generation have seen tremendous growth in the past decade [5], but due to their intermittency, a corresponding growth in grid-scale energy storage is needed to truly achieve grid decarbonization. To achieve profitability and encourage investment in energy storage infrastructure, storage operators use forecasts of electricity prices to schedule battery charging/discharging to maximize profit, and they rely on uncertainty estimates to minimize financial and operational risk. A common method used by battery storage operators to make decisions under uncertainty is robust optimization [21, 14]. The usual approach known as "estimate then optimize", separates the decision-making problem into two distinct stages. In the "estimate" stage, a predictive model is trained to forecast electricity prices, yielding an uncertainty set around future prices. Then, in the "optimize" stage, the forecast uncertainty set is used as a parameter in a robust optimization problem to output the battery charge/discharge schedule. Notably, any cost or loss associated with this downstream decision is usually not provided as feedback to the predictive model.

While a recent line of work [7, 16, 20, 8] has made steps toward bridging the gap between uncertainty quantification and robust optimization-driven decision-making, existing approaches are suboptimal for several reasons: **(1) The predictive model is not trained with feedback from the downstream objective**, which limits the model's performance on the decision-making task loss; **(2) For the robust optimization to be tractable, the forecast uncertainty sets have restricted parametric forms.** Common parametric forms include box and ellipsoidal uncertainty sets, limiting the expressivity of

[*]denotes equal contribution

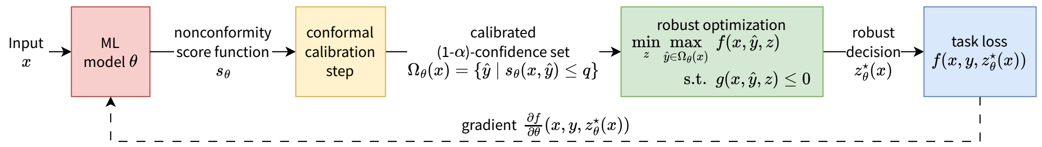
Input x
ML model θ
nonconformity score function s_θ
conformal calibration step
calibrated (1-α)-confidence set $\Omega_\theta(x) = \{\hat{y} \mid s_\theta(x, \hat{y}) \le q\}$
robust optimization $\min_z \max_{\hat{y} \in \Omega_\theta(x)} f(x, \hat{y}, z)$ s.t. $g(x, \hat{y}, z) \le 0$
robust decision $z^\star_\theta(x)$
task loss $f(x, y, z^\star_\theta(x))$
gradient $\frac{\partial f}{\partial \theta}(x, y, z^\star_\theta(x))$

Figure 1: Our proposed framework for end-to-end conformal calibration for optimization under uncertainty updates the machine learning model using gradients from the task loss.

uncertainty estimates; and **(3) Because neural network models are often poor at estimating their own uncertainty, the forecasts may not be well-calibrated.** Recent approaches such as isotonic regression [13] and conformal prediction [17] have made progress in providing *calibrated* uncertainty estimates from deep learning models, but such methods are typically applied post-hoc to pre-trained models and are therefore difficult to incorporate into an end-to-end training procedure.

In this work, we make three specific contributions addressing each issue identified above. First, **we develop a methodology for training prediction models end-to-end with downstream decision-making objectives and conformally calibrated uncertainty estimates for the *conditional robust optimization* problem.** Following our initial method proposal [22], we include differentiable conformal calibration layers in our model during training. This closes the feedback loop, ensuring that the uncertainty's impact on the downstream objective is accounted for during training, since not all model errors nor uncertainty estimates will result in the same downstream cost. Second, **we use partially input-convex neural networks (PICNNs) [3] to tractably approximate arbitrary convex uncertainty sets for the conditional robust optimization problem.** Due to the universal convex function approximation property of ICNNs [6], this approach enables training far more general representations of uncertainty than prior works have considered, which in turn yields improvements on downstream decision-making performance. Finally, **we propose an exact and computationally efficient method for differentiating through the conformal prediction procedure during training.** In contrast to prior work [18], our method gives exact gradients, without relying on approximate ranking and sorting methods.

We empirically evaluate our approach on an energy storage arbitrage task and demonstrate conclusively that the combination of end-to-end training with the flexibility of the PICNN-based uncertainty sets achieves substantial improvement over baseline methods. Our code is available on GitHub.[2]

2 Problem Statement and Background

Our problem setting is defined formally as follows: suppose that data $(x, y) \in \mathbb{R}^m \times \mathbb{R}^n$ is sampled i.i.d. from an unknown joint distribution $\mathcal{P}$. Upon observing the input x (but not the label y), an agent makes a decision $z \in \mathbb{R}^p$. After the decision is made, the true label y is revealed, and the agent incurs a *task loss* $f(x, y, z)$, for some known task loss function $f : \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}$. In addition, the agent's decision must satisfy a set of joint constraints $g(x, z) \le 0$.

Concretely, we consider the energy storage problem posed by [10], where a grid-scale battery operator predicts electricity prices $y \in \mathbb{R}^T$ over a T-step horizon and uses the prediction to decide on a battery charge/discharge schedule $z = (z^{\text{in}}, z^{\text{out}}, z^{\text{state}})$ for price arbitrage. The vector $z^{\text{in}} \in \mathbb{R}^T$ is the amount to charge, $z^{\text{out}} \in \mathbb{R}^T$ is the amount to discharge, and $z^{\text{state}} \in \mathbb{R}^T$ is the battery's state of charge. The input features x include the past day's prices and temperature, the next day's energy load forecast and temperature forecast, binary indicators of weekends or holidays, and yearly sinusoidal features. The battery has capacity B, charging efficiency γ, and maximum charge/discharge rates c^{in} and c^{out}. The task loss function f represents the multiple objectives of maximizing profit, flexibility to participate in the ancillary services market by keeping the battery near half its capacity (with weight λ), and battery health by discouraging rapid charging/discharging (with weight ϵ):

$$f(y, z) = \sum_{t=1}^{T} y_t (z^{\text{in}} - z^{\text{out}})_t + \lambda \left\| z^{\text{state}} - \frac{B}{2} \mathbf{1} \right\|^2 + \epsilon \left\| z^{\text{in}} \right\|^2 + \epsilon \left\| z^{\text{out}} \right\|^2 . \qquad (1)$$

[2] https://github.com/chrisyeh96/e2e-conformal

The constraints are given by

$$\begin{aligned} &z_0^{\text{state}} = B/2, \qquad z_t^{\text{state}} = z_{t-1}^{\text{state}} - z_t^{\text{out}} + \gamma z_t^{\text{in}} \qquad \forall t = 1, \dots, T \\ &0 \le z^{\text{in}} \le c^{\text{in}}, \qquad 0 \le z^{\text{out}} \le c^{\text{out}}, \qquad 0 \le z^{\text{state}} \le B. \end{aligned} \tag{2}$$

Because the agent does not observe the label y prior to making its decision, ensuring good performance and constraint satisfaction requires that the agent makes decisions z that are *robust* to the various outcomes of y. A common objective is to choose z to robustly minimize the task loss and satisfy the constraints over all realizations of y within a $(1-\alpha)$-confidence region $\Omega(x) \subset \mathbb{R}^n$ of the true conditional distribution $\mathcal{P}(y \mid x)$, where $\alpha \in (0,1)$ is a fixed risk level chosen based on operational requirements. In this case, the agent's robust decision can be expressed as the optimal solution to the following **conditional robust optimization (CRO)** problem [7, 8]:

$$z^\star(x) := \arg\min_{z \in \mathbb{R}^p} \max_{\hat{y} \in \Omega(x)} f(x, \hat{y}, z) \quad \text{s.t. } g(x, z) \le 0. \tag{3}$$

After the agent decides $z^\star(x)$, the true label y is revealed, and the agent incurs the task loss $f(x, y, z^\star(x))$. Thus, the agent seeks to minimize the expected task loss $\mathbb{E}_{(x,y)\sim\mathcal{P}}\left[f(x, y, z^\star(x))\right]$.

While the joint distribution $\mathcal{P}$ is unknown, we assume that we have access to a dataset $D = \{(x_i, y_i)\}_{i=1}^N$ of samples from $\mathcal{P}$. Then, our objective is to train a machine learning model to learn an approximate $(1-\alpha)$-confidence set $\Omega(x)$ of possible y values for each input x. Formally, our learned $\Omega(x)$ should satisfy the *marginal coverage* guarantee: $\mathbb{P}_{(x,y)\sim\mathcal{P}}\left(y \in \Omega(x)\right) \ge 1 - \alpha$.

3 Methodology: Uncertainty Calibration, Representation, and Training

Here, we describe our method for end-to-end task-aware training of predictive models with conformally calibrated uncertainty for the CRO problem (3). Figure 1 illustrates our framework, and Algorithm 1 (in Appendix B) shows pseudocode for training and inference. Our overarching goal is to learn uncertainty sets $\Omega(x)$ which provide $(1-\alpha)$ coverage for any choice of $\alpha \in (0,1)$, and which offer the lowest possible task loss. There are three primary questions that we must consider to this end; we briefly address each of these questions in turn, with details delegated to Appendix C due to space constraints.

1. How can we guarantee that the uncertainty set $\Omega(x)$ provides coverage at level $1-\alpha$?
2. How should the uncertainty set $\Omega(x)$ be parametrized?
3. How can the uncertainty set $\Omega(x)$ be learned to minimize the agent's expected task loss?

Conformal uncertainty set calibration Suppose that the uncertainty set is represented in the form $\Omega(x) = \{\hat{y} \in \mathbb{R}^n \mid s(x, \hat{y}) \le q\}$, where $s : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ is an arbitrary *nonconformity score function* and $q \in \mathbb{R}$ is a scalar. We use the split conformal prediction procedure [4] at inference for choosing q to ensure that $\Omega(x)$ provides marginal coverage at any confidence level $1-\alpha$. (During training, we apply a separate differentiable conformal prediction procedure, as discussed below.)

Representations of the uncertainty set In general, the uncertainty set $\Omega(x)$ must be convex for the robust optimization problem (3) to be tractable. To allow $\Omega(x)$ to approximate any arbitrary convex uncertainty set, we parametrize $\Omega_\theta(x)$ via a sublevel set of a general partially-convex function $s_\theta : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$, which is convex only in the second input vector. Then, fixing x, any $\Omega_\theta(x) = \{\hat{y} \in \mathbb{R}^n \mid s_\theta(x, \hat{y}) \le q\}$ is a q-sublevel set of $s_\theta(x, \cdot)$ and is therefore convex. In essence, we propose directly learning the nonconformity score function for conformal calibration using a PICNN [3]. Assuming that the task loss f is convex in z and bilinear in (y, z) (this assumption is satisfied by (1)), we can take the dual of the inner maximization problem to transform the robust optimization problem (3) into an equivalent tractable convex non-robust problem.

End-to-end training and calibration Our end-to-end training uses minibatch gradient descent to minimize the empirical task loss $\ell(\theta) = \frac{1}{N}\ell_i(\theta)$ where $\ell_i(\theta) = f(x_i, y_i, z_\theta^\star(x_i))$. This requires differentiating through both the robust optimization problem as well as the conformal prediction step. The gradient is $\frac{\mathrm{d}\ell_i}{\mathrm{d}\theta} = \frac{\partial f}{\partial z}\big|_{(x_i, y_i, z_\theta^\star(x_i))} \frac{\partial z_\theta^\star}{\partial \theta}\big|_{x_i}$, where $\frac{\partial z_\theta^\star}{\partial \theta}\big|_{x_i}$ can be computed by differentiating through the Karush–Kuhn–Tucker (KKT) conditions of the optimization problem [2]. To include calibration during training, we take inspiration from the conformal training approach [18] in which a separate q is chosen in each minibatch, as shown in Algorithm 1. The chosen q depends on θ (through s_θ), and $z_\theta^\star(x_i)$ depends on the chosen q. Therefore $\frac{\partial z_\theta^\star}{\partial \theta}$ involves calculating $\frac{\partial z_\theta^\star}{\partial q}\frac{\partial q}{\partial \theta}$, where

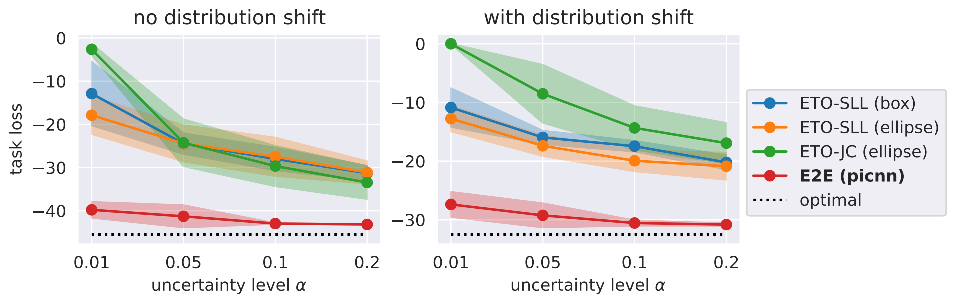

Figure 2: Average test set task loss (mean ± 1 stddev across 10 runs) at different uncertainty levels α. The box and ellipsoidal uncertainty set methods from [19] are denoted “ETO-SLL”, whereas the ellipsoidal uncertainty set method from [11] is denoted “ETO-JC.” The dashed-line “optimal” represents the best task loss achievable with perfect foresight of future electricity prices. Lower values are better. Our method **E2E (picnn)** outperforms all baseline methods and approaches the optimal.

$\frac{\partial q}{\partial \theta}$ requires differentiating through the empirical quantile function. Whereas [18] uses a smoothed approximate quantile function for calculating q, we find the smoothing unnecessary, as the gradient of the empirical quantile function is unique and well-defined almost everywhere.

4 Results and Conclusion

Experiments We compare our end-to-end method with PICNN-based uncertainty sets on the battery storage problem described in Section 2 against two-stage estimate-then-optimize (ETO) baseline methods [11, 19] that use more restrictive box and ellipsoidal uncertainty sets. Following [10], we set $T = 24$ hours, $B = 1$, $\gamma = 0.9$, $c^{\text{in}} = 0.5$, $c^{\text{out}} = 0.2$, $\lambda = 0.1$, and $\epsilon = 0.05$, and we use actual 2011-16 electricity price data and load forecasts from the PJM day-ahead market. We test our method both without distribution shift (where our training and test sets were sampled uniformly at random, thus ensuring exchangeability and guaranteeing marginal coverage) and on the more realistic setting with distribution shift by splitting our data temporally (where the training set is the first 80% of days and the test set is the last 20% of days). As shown in Figure 2, which plots average test set task loss at different levels of uncertainty α, our proposed method conclusively outperforms the previous baseline approaches at all uncertainty levels, with and without distribution shift.

Limitations We recognize that our energy storage problem ((1) and (2), originally posed in [10]) is a significantly simplified version of actual battery operations, which often involve participation in multiple energy markets and significantly more constraints. Moreover, our framework requires various assumptions on the functions f and g in the conditional robust optimization problem (3) in order to ensure tractability, such as convexity and bilinearity in (y, z). Lastly, while the PICNN uncertainty representation can represent general convex uncertainties, it cannot handle more general nonconvex uncertainty regions. Overcoming these assumptions would be an interesting direction for future work.

Pathway to Climate Impact As mentioned in the Introduction, improving energy storage operations is paramount for decarbonizing the electricity grid to accommodate increasing intermittent renewable generation. Due to the strong performance of our approach, we have already engaged in discussions with multiple companies that use AI in energy operations to identify opportunities to deploy our general end-to-end framework in real energy systems. With some modifications, we believe that our method may also be adapted for other related smart grid and grid decarbonization applications, such as carbon-aware EV charging and demand response.

Acknowledgments and Disclosure of Funding

We thank Priya Donti for helpful discussions. The authors acknowledge support from an NSF Graduate Research Fellowship (DGE-2139433); NSF Grants CNS-2146814, CPS-2136197, CNS-2106403, and NGSDI-2105648; Amazon AWS; and the Caltech Resnick Sustainability Institute.

References

[1] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter. Differentiable Convex Optimization Layers. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[2] B. Amos and J. Z. Kolter. OptNet: Differentiable Optimization as a Layer in Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 136–145. PMLR, July 2017. ISSN: 2640-3498.

[3] B. Amos, L. Xu, and J. Z. Kolter. Input Convex Neural Networks, June 2017. arXiv:1609.07152 [cs, math].

[4] A. N. Angelopoulos and S. Bates. Conformal Prediction: A Gentle Introduction. *Foundations and Trends® in Machine Learning*, 16(4):494–591, 2023.

[5] K. Antonio. Renewable generation surpassed coal and nuclear in the U.S. electric power sector in 2022, Mar. 2023.

[6] Y. Chen, Y. Shi, and B. Zhang. Optimal Control Via Neural Networks: A Convex Approach. In *International Conference on Learning Representations*, 2019.

[7] A. R. Chenreddy, N. Bandi, and E. Delage. Data-Driven Conditional Robust Optimization. In *Advances in Neural Information Processing Systems*, volume 35, pages 9525–9537, Dec. 2022.

[8] A. R. Chenreddy and E. Delage. End-to-end Conditional Robust Optimization. In *Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence*, pages 736–748. PMLR, Sept. 2024. ISSN: 2640-3498.

[9] S. Diamond and S. Boyd. CVXPY: A Python-Embedded Modeling Language for Convex Optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[10] P. L. Donti, B. Amos, and J. Z. Kolter. Task-based End-to-end Model Learning in Stochastic Optimization. In *Advances in Neural Information Processing Systems*, volume 30, Long Beach, CA, USA, Dec. 2017. Curran Associates, Inc. arXiv:1703.04529 [cs].

[11] C. Johnstone and B. Cox. Conformal uncertainty sets for robust optimization. In *Proceedings of the Tenth Symposium on Conformal and Probabilistic Prediction and Applications*, pages 72–90. PMLR, Sept. 2021. ISSN: 2640-3498.

[12] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, San Diego, CA, USA, May 2015.

[13] V. Kuleshov, N. Fenner, and S. Ermon. Accurate Uncertainties for Deep Learning Using Calibrated Regression. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2796–2804. PMLR, July 2018. ISSN: 2640-3498.

[14] S. S. Parvar and H. Nazaripouya. Optimal Operation of Battery Energy Storage Under Uncertainty Using Data-Driven Distributionally Robust Optimization. *Electric Power Systems Research*, 211:108180, Oct. 2022.

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: an imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, number 721, pages 8026–8037. Curran Associates Inc., Red Hook, NY, USA, Dec. 2019.

[16] Y. P. Patel, S. Rayan, and A. Tewari. Conformal Contextual Robust Optimization. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, pages 2485–2493. PMLR, Apr. 2024. ISSN: 2640-3498.

[17] G. Shafer and V. Vovk. A Tutorial on Conformal Prediction. *Journal of Machine Learning Research*, 9(12):371–421, 2008.

[18] D. Stutz, Krishnamurthy, Dvijotham, A. T. Cemgil, and A. Doucet. Learning Optimal Conformal Classifiers, May 2022. arXiv:2110.09192 [cs, stat].

[19] C. Sun, L. Liu, and X. Li. Predict-then-Calibrate: A New Perspective of Robust Contextual LP. Nov. 2023.

[20] I. Wang, C. Becker, B. Van Parys, and B. Stellato. Learning Decision-Focused Uncertainty Sets in Robust Optimization, July 2024. arXiv:2305.19225 [math].

[21] X. Yan, C. Gu, X. Zhang, and F. Li. Robust Optimization-Based Energy Storage Operation for System Congestion Management. *IEEE Systems Journal*, 14(2):2694–2702, June 2020.

[22] C. Yeh, N. Christianson, S. Low, A. Wierman, and Y. Yue. Decision-aware uncertainty-calibrated deep learning for robust energy system operation. In *ICLR 2023 Workshop on Tackling Climate Change with Machine Learning*. Climate Change AI, May 2023.

A Additional experimental results

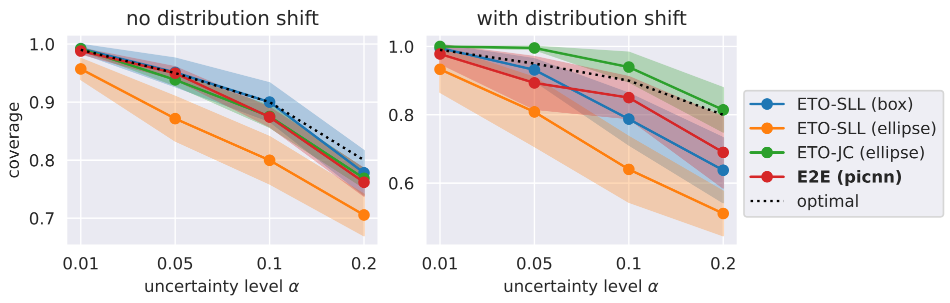

Figure A1: Average test set coverage (mean ± 1 stddev across 10 runs) at different uncertainty levels α. The box and ellipsoidal uncertainty set methods from [19] are denoted "ETO-SLL", whereas the ellipsoidal uncertainty set method from [11] is denoted "ETO-JC." The dashed-line "optimal" represents the desired marginal coverage corresponding to each uncertainty level α. Our method **E2E (picnn)** has coverage that is close to the target level.

The test set coverage obtained by the learned uncertainty sets is plotted in Figure A1. Most models (including our E2E approach) obtain coverage close to the target level, confirming that the improvements in task loss performance from our E2E approach do not come at the cost of worse coverage.

B Algorithm pseudocode

Algorithm 1 End-to-end conformal calibration for robust decisions under uncertainty

```
function TRAIN(training data $D = \{(x_i, y_i)\}_{i=1}^N$, uncertainty level $\alpha$, initial model parameters $\theta$)
    for mini-batch $B \subset \{1, \dots, N\}$ do
        Randomly split batch: $B = (B_{\text{cal}}, B_{\text{pred}})$
        Compute $q = \text{QUANTILE}(\{s_\theta(x_i, y_i)\}_{i \in B_{\text{cal}}},\ 1 - \alpha)$
        for $i \in B_{\text{pred}}$ do
            Solve for robust decision $z_\theta^\star(x_i)$ using (6)
            Compute gradient of task loss: $d\theta_i = \partial f(x_i, y_i, z_\theta^\star(x_i)) / \partial\theta$
        Update $\theta$ using gradients $\sum_{i \in B_{\text{pred}}} d\theta_i$

function INFERENCE(model parameters $\theta$, calibration data $D_{\text{cal}} = \{(x_i, y_i)\}_{i=1}^M$, uncertainty level $\alpha$, input $x$)
    Compute $q = \text{QUANTILE}(\{s_\theta(\tilde{x}, \tilde{y})\}_{(\tilde{x}, \tilde{y}) \in D_{\text{cal}}},\ 1 - \alpha)$
    return robust decision $z_\theta^\star(x)$ using (6)

function QUANTILE(scores $S = \{s_i\}_{i=1}^M$, level $\beta$)
    $s_{(1)}, \dots, s_{(M+1)} = \text{SORTASCENDING}(S \cup \{+\infty\})$        ▷ does not need to be differentiable
    return $s_{(\lceil (M+1)\beta \rceil)}$
```

C PICNN uncertainty set representation and calibration

This section gives more details for the discussion from Section 3. First, we formally define the marginal coverage guarantee that we would like to satisfy.

Definition 1 (marginal coverage)**.** An uncertainty set $\Omega(x)$ for the distribution $\mathcal{P}$ provides *marginal coverage at level* $(1 - \alpha)$ if $\mathbb{P}_{(x,y)\sim\mathcal{P}}\left(y \in \Omega(x)\right) \geq 1 - \alpha$.

Next, we make the following assumptions on the functions f and g to ensure tractability of the resulting optimization problem.

Assumptions. We assume that $g(x, y, z) = g(x, z)$ does not depend on y, and that the task loss has the form $f(x, y, z) = y^\top F z + \tilde{f}(x, z)$ for some matrix $F \in \mathbb{R}^{n \times p}$ and auxiliary function

$\tilde{f} : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$—that is, the task loss decomposes into a function $\tilde{f}(x, z)$ independent of y and a bilinear term $y^\top Fz$. We further assume that $g(x, z)$ and $\tilde{f}(x, z)$ are convex in z.

C.1 Conformal uncertainty set calibration

Suppose that the uncertainty set is represented in the form

$$\Omega(x) = \{\hat{y} \in \mathbb{R}^n \mid s(x, \hat{y}) \le q\}, \tag{4}$$

where $s : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ is an arbitrary *nonconformity score function* and $q \in \mathbb{R}$. We use the split conformal prediction procedure [4] at inference for choosing q to ensure that $\Omega(x)$ provides marginal coverage at any confidence level $1 - \alpha$. (During training, we apply a separate differentiable conformal prediction procedure, as discussed in Appendix C.3.)

Lemma 1 (from [4], Appendix D)**.** Let $D_{\text{cal}} = \{(x_i, y_i)\}_{i=1}^M$ be a calibration dataset drawn exchangeably (*e.g.*, i.i.d.) from $\mathcal{P}$, and let $s_i = s_\theta(x_i, y_i)$. If $q = \textsc{Quantile}(\{s_i\}_{i=1}^M,\ 1 - \alpha)$ (see Algorithm 1) is the empirical $(1 - \alpha)$-quantile of the set $\{s_i\}_{i=1}^M$ and (x, y) is drawn exchangeably with D_{cal}, then $\Omega_\theta(x)$ has the coverage guarantee

$$1 - \alpha \le \mathbb{P}_{x,y,D_{\text{cal}}}(y \in \Omega_\theta(x)) \le 1 - \alpha + \frac{1}{M+1}.$$

We use split conformal prediction, rather than full conformal prediction, both for computational tractability and to avoid the problem of nonconvex uncertainty sets that can arise from the full conformal approach, as noted in [11]. For the rest of this paper, we assume $\alpha \in [\frac{1}{M+1}, 1)$ so that $q = \textsc{Quantile}(\{s_i\}_{i=1}^M,\ 1 - \alpha) < \infty$ is finite. Thus, for appropriate choices of the nonconformity score function s_θ, the uncertainty set $\Omega_\theta(x)$ is not unbounded.

C.2 Representations of the uncertainty set

In general, the uncertainty set $\Omega(x)$ must be convex in order for the robust optimization problem (3) to be tractable. Therefore, we aim to let $\Omega(x)$ approximate any arbitrary convex uncertainty set by representing $\Omega(x)$ via a sublevel set of a partially-convex function $s_\theta : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$, which is convex only in the second input vector. Then, fixing x, any $\Omega_\theta(x) = \{\hat{y} \in \mathbb{R}^n \mid s_\theta(x, \hat{y}) \le q\}$ is a q-sublevel set of $s_\theta(x, \cdot)$ and is therefore a convex set. In essence, we propose directly learning the nonconformity score function for conformal calibration.

A natural model for learning the function s_θ is a partially input-convex neural network (PICNN) [3], which can approximate any partially-convex function [6]. We consider a PICNN defined as $s_\theta(x, y) = W_L \sigma_L + V_L y + b_L$, where

$$\begin{aligned}
\sigma_0 &= \mathbf{0}, \qquad u_0 = x & W_l &= \bar{W}_l \operatorname{diag}([\hat{W}_l u_l + w_l]_+) && \text{(5a)}\\
u_{l+1} &= \operatorname{ReLU}(R_l u_l + r_l) & V_l &= \bar{V}_l \operatorname{diag}(\hat{V}_l u_l + v_l) && \text{(5b)}\\
\sigma_{l+1} &= \operatorname{ReLU}(W_l \sigma_l + V_l y + b_l) & b_l &= \bar{B}_l u_l + \bar{b}_l. && \text{(5c)}
\end{aligned}$$

The weights of the neural network are $\theta = (R_l, r_l, \bar{W}_l, \hat{W}_l, w_l, \bar{V}_l, \hat{V}_l, v_l, \bar{B}_l, \bar{b}_l)_{l=0}^L$, and the matrices $\bar{W}_l$ are constrained to be entrywise nonnegative to ensure convexity of s_θ with respect to y. For ease of notation, we assume all hidden layers $\sigma_1, \dots, \sigma_L$ have the same dimension d.

Let $\Omega_\theta(x) = \{\hat{y} \in \mathbb{R}^n \mid s_\theta(x, \hat{y}) \le q\}$ where $q \in \mathbb{R}$ is chosen by the split conformal procedure detailed in Appendix C.1. Then, by taking the dual of the inner maximization problem (see Appendix D.1), we can transform the robust optimization problem (3) into an equivalent non-robust problem

$$\begin{aligned}
z_\theta^\star(x) = \arg\min_{z \in \mathbb{R}^p} \min_{\nu \in \mathbb{R}^{2Ld+1}} \quad & b(\theta, q)^\top \nu + \tilde{f}(x, z)\\
\text{s.t.} \quad & A(\theta)^\top \nu = \begin{bmatrix} Fz \\ \mathbf{0} \end{bmatrix}, \quad \nu \ge \mathbf{0}, \quad g(x, z) \le 0
\end{aligned} \tag{6}$$

where $A(\theta) \in \mathbb{R}^{(2Ld+1)\times(n+Ld)}$ and $b(\theta, q) \in \mathbb{R}^{2Ld+1}$ are constructed from the weights θ of the PICNN, and b also depends on q. Finally, observe that if $\tilde{f}(x, z)$ and $g(x, z)$ are convex in z, then the optimization problem (6) is convex.

In some cases during training, the inner maximization problem of (3) with PICNN-parametrized uncertainty set may be infeasible (if too small a q was chosen by the split conformal procedure and $\Omega_\theta(x)$ is empty). This will lead, respectively, to an infeasible or unbounded equivalent problem (6). We can avoid this concern by solving the following convex optimization problem

$$q_{\min} = \min_{\hat{y} \in \mathbb{R}^n} s_\theta(x, \hat{y})$$

to calculate the minimum value of the PICNN. In the case that the q chosen by the split conformal procedure is smaller than $q_{\min}$, we simply replace q with $q_{\min}$; since in this case we are replacing q with a larger value, this modification preserves the marginal coverage guarantee for the uncertainty set $\Omega_\theta(x)$.

C.3 End-to-end training and calibration

Our end-to-end training uses minibatch gradient descent to minimize the empirical task loss $\ell(\theta) = \frac{1}{N}\ell_i(\theta)$ where $\ell_i(\theta) = f(x_i, y_i, z^\star_\theta(x_i))$. This requires differentiating through both the robust optimization problem as well as the conformal prediction step. The gradient is $\frac{\mathrm{d}\ell_i}{\mathrm{d}\theta} = \frac{\partial f}{\partial z}|_{(x_i, y_i, z^\star_\theta(x_i))} \frac{\partial z^\star_\theta}{\partial \theta}|_{x_i}$, where $\frac{\partial z^\star_\theta}{\partial \theta}|_{x_i}$ can be computed by differentiating through the Karush–Kuhn–Tucker (KKT) conditions of the convex optimization problem following the approach of [2], under mild assumptions on the differentiability of f and g.

To include calibration during training, we take inspiration from the conformal training approach [18] in which a separate q is chosen in each minibatch, as shown in Algorithm 1. The chosen q depends on θ (through s_θ), and $z^\star_\theta(x_i)$ depends on the chosen q. Therefore $\frac{\partial z^\star_\theta}{\partial \theta}$ involves calculating $\frac{\partial z^\star_\theta}{\partial q}\frac{\partial q}{\partial \theta}$, where $\frac{\partial q}{\partial \theta}$ requires differentiating through the empirical quantile function. Whereas [18] uses a smoothed approximate quantile function for calculating q, we find the smoothing unnecessary, as the gradient of the empirical quantile function is unique and well-defined almost everywhere.

After training has concluded and we have performed the final conformal calibration step, the resulting model enjoys the following theoretical guarantee on performance (*cf.* [19, Proposition 1]).

Proposition 1. After following our training and calibration procedure, we achieve with probability $1-\alpha$ (with respect to x, y, and the calibration set D_{cal}) the following upper bound on task loss:

$$f(x, y, z^\star(x)) \leq \min_{z \in \mathbb{R}^p} \max_{\hat{y} \in \Omega_\theta(x)} f(x, \hat{y}, z) \qquad \text{s.t.} \qquad g(x, \hat{y}, z) \leq 0.$$

This result is an immediate consequence of the split conformal coverage guarantee from Lemma 1, which ensures that for the true $(x, y) \sim \mathcal{P}$, $\mathbb{P}_{x,y,D_{\text{cal}}}(y \in \Omega(x)) \geq 1-\alpha$. The realized task loss will, with probability $1-\alpha$, therefore improve upon the optimal value of the robust problem (3). Moreover, this guarantee holds despite the fact that the distribution $\mathcal{P}(y \mid x)$ is unknown, and the dependence of the learned uncertainty set $\Omega_\theta(x)$ on x allows for taking advantage of heteroskedasticity that, together with our end-to-end training framework, yield substantial improvements in average-case performance in conjunction with our the robust guarantee offered by Proposition 1.

D Maximizing over a PICNN uncertainty set

We consider robust optimization problems of the form

$$\min_{z \in \mathbb{R}^p} \max_{\hat{y} \in \mathbb{R}^n} \hat{y}^\top F z + \tilde{f}(x, z) \qquad \text{s.t.} \qquad \hat{y} \in \Omega(x), \quad g(x, z) \leq 0.$$

For fixed z, the inner maximization problem is

$$\max_{\hat{y} \in \mathbb{R}^n} \hat{y}^\top F z \qquad \text{s.t.} \qquad \hat{y} \in \Omega(x),$$

which we analyze in the more abstract form

$$\max_{y \in \mathbb{R}^n} c^\top y \qquad \text{s.t.} \qquad y \in \Omega$$

for arbitrary $c \in \mathbb{R}^n \setminus \{0\}$. The subsections of this appendix derive the dual form of this maximization problem for specific representations of the uncertainty set Ω.

Suppose y is standardized or whitened by an affine transformation with $\mu \in \mathbb{R}^n$ and invertible matrix $W \in \mathbb{R}^{n \times n}$

$$y_{\text{transformed}} = W^{-1}(y - \mu)$$

so that Ω is an uncertainty set on the transformed $y_{\text{transformed}}$. Then, the original primal objective can be recovered as

$$c^\top y = c^\top (W y_{\text{transformed}} + \mu) = (Wc)^\top y_{\text{transformed}} + c^\top \mu.$$

In our experiments, we use element-wise standardization of y by setting $W = \operatorname{diag}(y_{\text{std}})$, where $y_{\text{std}} \in \mathbb{R}^n$ is the element-wise standard-deviation of y.

D.1 Maximizing over sublevel set of PICNN

Let $s_\theta : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ be a partially input-convex neural network (PICNN) with ReLU activations as described in (5), so that $s_\theta(x, y)$ is convex in y. Suppose that all the hidden layers have the same dimension d (*i.e.*, $\forall l = 0, \dots, L-1$: $W_l \in \mathbb{R}^{d \times d}$, $V_l \in \mathbb{R}^{d \times n}$, $b_l \in \mathbb{R}^d$), and the final layer L has $W_L \in \mathbb{R}^{1 \times d}$, $V_L \in \mathbb{R}^{1 \times n}$, $b_L \in \mathbb{R}$. Let $c \in \mathbb{R}^n$ be any vector. Then, the optimization problem

$$\max_{y \in \mathbb{R}^n} \; c^\top y \qquad \text{s.t.} \qquad s_\theta(x, y) \leq q \tag{7}$$

can be equivalently written as

$$\begin{aligned}
\max_{y \in \mathbb{R}^n,\, \sigma_1, \dots, \sigma_L \in \mathbb{R}^d} \quad & c^\top y && && (8a)\\
\text{s.t.} \quad & \sigma_l \geq \mathbf{0}_d && \forall l = 1, \dots, L && (8b)\\
& \sigma_{l+1} \geq W_l \sigma_l + V_l y + b_l && \forall l = 0, \dots, L-1 && (8c)\\
& W_L \sigma_L + V_L y + b_L \leq q, && && (8d)
\end{aligned}$$

To see that this is the case, first note that (8) is a relaxed form of (7), obtained by replacing the equalities $\sigma_{l+1} = \operatorname{ReLU}(W_l \sigma_l + V_l y + b_l)$ in the definition of the PICNN (5) with the two separate inequalities $\sigma_{l+1} \geq \mathbf{0}_d$ and $\sigma_{l+1} \geq W_l \sigma_l + V_l y + b_l$ for each $l = 0, \dots, L-1$. As such, the optimal value of (8) is no less than that of (7). However, given an optimal solution $y, \sigma_1, \dots, \sigma_L$ to (8), it is possible to obtain another feasible solution $y, \hat{\sigma}_1, \dots, \hat{\sigma}_L$ with the same optimal objective value by iteratively decreasing each component of σ_l until one of the two inequality constraints (8b), (8c) is tight, beginning at $l = 1$ and incrementing l once all entries of σ_l cannot be decreased further. This procedure of decreasing the entries in each σ_l will maintain problem feasibility, since the weight matrices W_l are all assumed to be entrywise nonnegative in the PICNN construction; in particular, this procedure will not increase the left-hand side of (8d). Moreover, since one of the two constraints (8b), (8c) will hold for each entry of each $\hat{\sigma}_l$, this immediately implies that y is feasible for the unrelaxed problem (7), and so (7) and (8) must have the same optimal value.

Having shown that we may replace the convex program (7) with a linear equivalent (8), we can write this latter problem in the matrix form

$$\max_{y \in \mathbb{R}^n,\, \sigma_1, \dots, \sigma_L \in \mathbb{R}^d} \; c^\top y \qquad \text{s.t.} \qquad A \begin{bmatrix} y \\ \sigma_1 \\ \vdots \\ \sigma_L \end{bmatrix} \leq b$$

where

$$A = \begin{bmatrix} & -I_d & & \\ & & \ddots & \\ & & & -I_d \\ V_0 & -I_d & & \\ \vdots & W_1 & \ddots & \\ \vdots & & \ddots & -I_d \\ V_L & & & W_L \end{bmatrix} \in \mathbb{R}^{(2Ld+1) \times (n+Ld)}, \qquad b = \begin{bmatrix} \mathbf{0}_d \\ \vdots \\ \mathbf{0}_d \\ -b_0 \\ \vdots \\ -b_{L-1} \\ q - b_L \end{bmatrix} \in \mathbb{R}^{2Ld+1}. \tag{9}$$

By strong duality, if this linear program has an optimal solution, its optimal value is equal to the optimal value of its dual problem:

$$\min_{\nu \in \mathbb{R}^{2Ld+1}} \; b^\top \nu \qquad \text{s.t.} \qquad A^\top \nu = \begin{bmatrix} c \\ \mathbf{0}_{Ld} \end{bmatrix}, \quad \nu \geq 0. \tag{10}$$

We can incorporate this dual problem (10) into the outer minimization of (3) to yield the non-robust form (6). For a more interpretable form of this dual problem, let $\nu^{(i)}$ denote the portion of the dual vector ν corresponding to the i-th block-row of matrix A, indexed from 0. That is, $\nu^{(i)} = \nu_{id+1:(i+1)d}$ for $i = 0, \dots, 2L-1$. Furthermore, let $\mu = \nu_{2Ld+1}$ be the last entry of ν. Written out, the dual problem (10) becomes

$$\begin{aligned}
\min_{\nu^{(0)},\dots,\nu^{(2L-1)} \in \mathbb{R}^d,\, \mu \in \mathbb{R}} \quad & \mu(q - b_L) - \sum_{l=0}^{L} b_l^\top \nu^{(L+l)} \\
\text{s.t.} \quad & \begin{bmatrix} V_0^\top & \cdots & V_L^\top \end{bmatrix} \nu_{Ld+1:} = c \\
& W_{l+1}^\top \nu^{(L+l+1)} - \nu^{(L+l)} - \nu^{(l)} = \mathbf{0}_d \qquad \forall l = 0, \dots, L-1 \\
& \nu \geq 0.
\end{aligned}$$

E Experiment details

Our experiments were conducted on a machine with two AMD EPYC 7513 32-Core processors, 1TiB RAM, and 4 NVIDIA A100 GPUs. However, we note that most of our experiments (including all of the end-to-end training) only used CPUs without any GPU acceleration.

We reserved 20% of the data for the test set. Of the remaining 80%, we further used a 80/20 split to create the training and validation sets.

Our PICNN has 2 hidden layers of 64 units each with ReLU activations. We used a batch size of 256 to train our models. We trained our models for a maximum of 100, with early stopping based on validation task loss. We used the Adam optimizer [12] with learning rate tuned over [1e-5, 1e-4, 1e-3] and L2 weight decay tuned over [0, 1e-4, 1e-3, 1e-2].

Our models were implemented using the PyTorch [15] and cvxpylayers [1] packages in Python. The optimization problem was implemented in cvxpy [9].