# Probabilistic Circuits That Know What They Don't Know (Supplementary Material)

**Fabrizio Ventola***[1] **Steven Braun***[1] **Zhongjie Yu**[1] **Martin Mundt**[1,2] **Kristian Kersting**[1,2,3,4]

[1]Department of Computer Science, TU Darmstadt, Darmstadt, Germany
[2]Hessian Center for AI (hessian.AI), Darmstadt, Germany
[3]German Research Center for Artificial Intelligence (DFKI), Darmstadt, Germany
[4]Centre for Cognitive Science, TU Darmstadt, Darmstadt, Germany

Our paper's supplementary material contains various supporting materials and complementary empirical evidence for our main paper's findings. Specifically, the appendix consists of six sections:

- **A Tractable dropout inference derivations:** We provide the full derivations behind the presented TDI equations of our main body's Section 3.
- **B Tractable dropout inference pseudocode algorithm:** We present the pseudocode algorithm of the TDI procedure.
- **C Experimental setup:** The section contains additional details with respect to the experimental setup for our empirical evaluations of the main body.
- **D PCs with TDI can detect OOD data:** We provide a complementary view on the ability of TDI in detecting OOD data, showing the predictive entropy and the predictive uncertainty of a PC with TDI.
- **E PCs with TDI are more robust to corruptions:** We present additional quantitative evidence in form of the remaining corruptions not shown in the main body to further support our findings that PCs with TDI are more robust on corrupted data.
- **F Societal impact:** We briefly discuss the societal impact of our contributions.

## A TRACTABLE DROPOUT INFERENCE DERIVATIONS

Appendix A contains the full set of derivations behind our closed-form solutions of TDI. Accordingly, the subsequent subsections follow the notation and structure of our main body's Section 3.

### A.1 MONTE CARLO DROPOUT UNCERTAINTY

With Monte Carlo dropout we can estimate model uncertainty by measuring the variance of the probability computed by the PC's root node. This is done by replacing all sum node computations with:

$$\mathsf{S} = \sum_i \delta_i w_i \mathsf{N}_i \, , \tag{1}$$

where $\delta_i \sim \text{Bern}(1-p)$ is the result of a Bernoulli trial with probability $1-p$. Here, $p$ is the dropout chance and $\mathsf{N}_i$ is the value of the $i$-th child. Note that the product node computations remain unchanged since they have no model parameters associated and simply compute factorizations. In the traditional Monte Carlo dropout applied to PCs, the forward pass for a single input is then repeated $L$ times to finally compute the expected value and variance of the root node (i.e., the PC

* indicates equal contribution

probability):

$$\mathbb{E}[\mathsf{N}_{\text{root}}] = \frac{1}{L}\sum_{i=1}^{L} \mathsf{N}_{\text{root},i} \tag{2}$$

$$\text{Var}[\mathsf{N}_{\text{root}}] = \frac{1}{L}\sum_{i=1}^{L} \left(\mathsf{N}_{\text{root},i} - \mathbb{E}[\mathsf{N}_{\text{root}}]\right)^2 \ , \tag{3}$$

where $\mathsf{N}_{\text{root},i}$ is the $i$-th random Monte Carlo trial of Eq. (1).

Since the Monte Carlo dropout formulation is based on repeated Bernoulli trials, we can now look at it from a different perspective. We can define $\mathsf{S}$ to be a function of the Bernoulli random variable $\delta_i$. As the expectation and variance for Bernoulli random variables are well known and easy to compute, we only need to formulate how they propagate bottom-up through the PC in a hierarchical manner, through sum and product nodes. We formally derive this in the next section.

## A.2 TDI'S ANALYTICAL SOLUTION TO DROPOUT UNCERTAINTY

To derive the analytical solution of expectation, variance, and covariance propagation for TDI, we make use of the basic rules of how the expectation, variance, and covariance behave under addition, multiplication, and constant scaling. Recall that the goal of the derivation is to express the expectation, variance, and covariance of a dropout node $\mathsf{N}_i$ in terms of the expectation, variance, and covariance of its children $\mathbb{E}[\mathsf{N}_j]$ , $\text{Var}[\mathsf{N}_j]$ , $\text{Cov}[\mathsf{N}_j, \mathsf{N}_{j'}]$ for $\mathsf{N}_j \in \text{children}(\mathsf{N}_i)$. This means that if we have a closed-form solution for the expectation, variance, and covariance of the leaf nodes, we can calculate the expectation, variance, and covariance of any arbitrary node in the PC by the bottom-up propagation of $\mathbb{E}[\mathsf{N}_j]$, $\text{Var}[\mathsf{N}_j]$, and $\text{Cov}[\mathsf{N}_j, \mathsf{N}_{j'}]$.

### A.2.1 Expectation: The Point Estimate

**Sum Nodes:** Using the linearity of the expectation, we can move the expectation into the sum and make use of the independence between the child nodes $\mathsf{N}_i$ and the Bernoulli RVs $\delta_i$ to extract $\mathbb{E}[\delta_i] = q$:

$$\begin{aligned}
\mathbb{E}[\mathsf{S}] &= \mathbb{E}\left[\sum_i \delta_i w_i \mathsf{N}_i\right] && (4)\\
&= \sum_i \mathbb{E}[\delta_i w_i \mathsf{N}_i] && (5)\\
&= \sum_i \mathbb{E}[\delta_i]\, w_i \mathbb{E}[\mathsf{N}_i] && (6)\\
&= \sum_i q w_i \mathbb{E}[\mathsf{N}_i] && (7)\\
&= q \sum_i w_i \mathbb{E}[\mathsf{N}_i] \quad . && (8)
\end{aligned}$$

As of Eq. (8), we can see that the expected outcome of the sum node is a convex combination with the original weights $w_i$ of the expected outcome of its children $\mathsf{N}_i$, scaled by the expected drop in likelihood $q$.

**Product Nodes:** The decomposability of product nodes ensures the independence of their children w.r.t. each other. This in turn leads to the product node expectation simply becoming the product over the expectations of its children, i.e.,

$$\mathbb{E}[\mathsf{P}] = \mathbb{E}\left[\prod_i \mathsf{N}_i\right] = \prod_i \mathbb{E}[\mathsf{N}_i] \quad . \tag{9}$$

### A.2.2 Variance: The Uncertainty Proxy

Similar to the original work of Monte Carlo dropout in neural networks of Gal and Ghahramani [2016], we have used the variance as the proxy for the uncertainty.

**Sum Nodes:** The sum node variance decomposes into a sum of two terms. The first term is based on the variances and expectations of its children and the second term accounts for the covariance between the combinations of all children:

$$\begin{aligned}
\mathrm{Var}[\mathsf{S}] &= \mathrm{Var}\left[\sum_i \delta_i w_i \mathsf{N}_i\right] && (10)\\
&= \sum_i \mathrm{Var}[\delta_i w_i \mathsf{N}_i] + \sum_{i \neq j} \mathrm{Cov}[\delta_i w_i \mathsf{N}_i, \delta_j w_j \mathsf{N}_j] && (11)\\
&= \sum_i w_i^2 \mathrm{Var}[\delta_i \mathsf{N}_i] + \sum_{i \neq j} w_i w_j \mathrm{Cov}[\delta_i \mathsf{N}_i, \delta_j \mathsf{N}_j] \quad . && (12)
\end{aligned}$$

For the first term of Eq. (12), we can further resolve $\mathrm{Var}[\delta_i \mathsf{N}_i]$ by employing the rules for the variance computation:

$$\begin{aligned}
\mathrm{Var}[\delta_i \mathsf{N}_i] &= \mathbb{E}\left[\delta_i^2\right] \mathbb{E}\left[\mathsf{N}_i^2\right] - \mathbb{E}[\delta_i]^2 \, \mathbb{E}[\mathsf{N}_i]^2 && (13)\\
&= q\mathbb{E}\left[\mathsf{N}_i^2\right] - q^2 \mathbb{E}[\mathsf{N}_i]^2 && (14)\\
&= q\left(\mathrm{Var}[\mathsf{N}_i] + \mathbb{E}[\mathsf{N}_i]^2\right) - q^2\mathbb{E}[\mathsf{N}_i]^2 && (15)\\
&= q\left(\mathrm{Var}[\mathsf{N}_i] + \mathbb{E}[\mathsf{N}_i]^2 - q\mathbb{E}[\mathsf{N}_i]^2\right) && (16)\\
&= q\left(\mathrm{Var}[\mathsf{N}_i] + (1-q)\,\mathbb{E}[\mathsf{N}_i]^2\right) && (17)\\
&= q\left(\mathrm{Var}[\mathsf{N}_i] + p\mathbb{E}[\mathsf{N}_i]^2\right) \quad . && (18)
\end{aligned}$$

The second term of Eq. (12), the summation over $\mathrm{Cov}[\delta_i \mathsf{N}_i, \delta_j \mathsf{N}_j]$, includes the covariance between all child nodes of $\mathsf{S}$:

$$\begin{aligned}
\mathrm{Cov}[\delta_i \mathsf{N}_i, \delta_j \mathsf{N}_j] &= \mathbb{E}[\delta_i \mathsf{N}_i \delta_j \mathsf{N}_j] - \mathbb{E}[\delta_i \mathsf{N}_i]\,\mathbb{E}[\delta_j \mathsf{N}_j] && (19)\\
&= \mathbb{E}[\delta_i]\,\mathbb{E}[\delta_j]\,\mathbb{E}[\mathsf{N}_i \mathsf{N}_j] - \mathbb{E}[\delta_i]\,\mathbb{E}[\mathsf{N}_i]\,\mathbb{E}[\delta_j]\,\mathbb{E}[\mathsf{N}_j] && (20)\\
&= q^2\left(\mathbb{E}[\mathsf{N}_i \mathsf{N}_j] - \mathbb{E}[\mathsf{N}_i]\,\mathbb{E}[\mathsf{N}_j]\right) && (21)\\
&= q^2 \mathrm{Cov}[\mathsf{N}_i, \mathsf{N}_j] \quad . && (22)
\end{aligned}$$

Thus, by putting the derivations of the two terms together, we obtain the expression for the variance of a sum node:

$$\mathrm{Var}[\mathsf{S}] = q\sum_i w_i^2\left(\mathrm{Var}[\mathsf{N}_i] + p\mathbb{E}[\mathsf{N}_i]^2\right) + q^2 \sum_{i \neq j} w_i w_j \mathrm{Cov}[\mathsf{N}_i, \mathsf{N}_j] \ . \qquad (23)$$

**Product Nodes:** Similarly to the case of a sum node, the product node variance decomposes into two product terms, by applying the product of independent variables rule, we obtain:

$$\begin{aligned}
\mathrm{Var}[\mathsf{P}] &= \mathrm{Var}\left[\prod_i \mathsf{N}_i\right] && (24)\\
&= \prod_i \mathbb{E}\left[\mathsf{N}_i^2\right] - \prod_i \mathbb{E}[\mathsf{N}_i]^2 && (25)\\
&= \prod_i \left(\mathrm{Var}[\mathsf{N}_i] + \mathbb{E}[\mathsf{N}_i]^2\right) - \prod_i \mathbb{E}[\mathsf{N}_i]^2 \quad . && (26)
\end{aligned}$$

### A.2.3 Covariance: The Evil

**Sum Nodes:** The covariance of two sum nodes, $\mathsf{S}^A$ and $\mathsf{S}^B$, neatly decomposes into a weighted sum of all covariance combinations of the sum node children, as shown in the following:

$$\begin{aligned}
\mathrm{Cov}\left[\mathsf{S}^A, \mathsf{S}^B\right] &= \mathrm{Cov}\left[\sum_i w_i^A \delta_i^A \mathsf{N}_i^A, \sum_j w_j^B \delta_j^B \mathsf{N}_j^B\right] && (27)\\
&= \mathbb{E}\left[\sum_i w_i^A \delta_i^A \mathsf{N}_i^A \sum_j w_j^B \delta_j^B \mathsf{N}_j^B\right] - \mathbb{E}\left[\sum_i w_i^A \delta_i^A \mathsf{N}_i^A\right] \mathbb{E}\left[\sum_j w_j^B \delta_j^B \mathsf{N}_j^B\right] && (28)\\
&= \sum_i w_i^A \sum_j w_j^B \mathbb{E}\left[\delta_i^A \mathsf{N}_i^A \delta_j^B \mathsf{N}_j^B\right] - \sum_i w_i^A \sum_j w_j^B \mathbb{E}\left[\delta_i^A \mathsf{N}_i^A\right] \mathbb{E}\left[\delta_j^B \mathsf{N}_j^B\right] && (29)\\
&= \sum_i w_i^A \sum_j w_j^B \mathbb{E}\left[\delta_i^A \delta_j^B\right] \mathbb{E}\left[\mathsf{N}_i^A \mathsf{N}_j^B\right] - \sum_i w_i^A \sum_j w_j^B \mathbb{E}\left[\delta_i^A\right] \mathbb{E}\left[\mathsf{N}_i^A\right] \mathbb{E}\left[\delta_j^B\right] \mathbb{E}\left[\mathsf{N}_j^B\right] && (30)\\
&= q^2 \left(\sum_i w_i^A \sum_j w_j^B \mathbb{E}\left[\mathsf{N}_i^A \mathsf{N}_j^B\right] - \sum_i w_i^A \sum_j w_j^B \mathbb{E}\left[\mathsf{N}_i^A\right] \mathbb{E}\left[\mathsf{N}_j^B\right]\right) && (31)\\
&= q^2 \left(\sum_i w_i^A \sum_j w_j^B \left(\mathbb{E}\left[\mathsf{N}_i^A \mathsf{N}_j^B\right] - \mathbb{E}\left[\mathsf{N}_i^A\right] \mathbb{E}\left[\mathsf{N}_j^B\right]\right)\right) && (32)\\
&= q^2 \sum_i w_i^A \sum_j w_j^B \mathrm{Cov}\left[\mathsf{N}_i^A, \mathsf{N}_j^B\right] \quad . && (33)
\end{aligned}$$

**Product Nodes:** As detailed in the paper's main body, we are unfortunately unable to provide a closed-form solution of the covariance between two product nodes for an arbitrary graph. This is due to the first expectation in the product node covariance:

$$\begin{aligned}
\mathrm{Cov}\left[\mathsf{P}^A, \mathsf{P}^B\right] &= \mathrm{Cov}\left[\prod_i \mathsf{N}_i^A, \prod_j \mathsf{N}_j^B\right] && (34)\\
&= \mathbb{E}\left[\prod_i \mathsf{N}_i^A \prod_j \mathsf{N}_j^B\right] - \mathbb{E}\left[\prod_i \mathsf{N}_i^A\right] \mathbb{E}\left[\prod_j \mathsf{N}_j^B\right] && (35)\\
&= \mathbb{E}\left[\prod_i \mathsf{N}_i^A \prod_j \mathsf{N}_j^B\right] - \prod_i \mathbb{E}\left[\mathsf{N}_i^A\right] \prod_j \mathbb{E}\left[\mathsf{N}_j^B\right] \quad . && (36)
\end{aligned}$$

Further resolving the term $\mathbb{E}\left[\prod_i \mathsf{N}_i^A \prod_j \mathsf{N}_j^B\right]$ without additional knowledge about specific node substructures is not possible. More specifically, pairs of $\mathsf{N}_i^A$ and $\mathsf{N}_j^B$ may not be independent and may share a common subset of nodes, deeper down in the PC structure. In this case, as alternatives, we have provided three possible solutions to solve Eq. (34) in the main body. For convenience, we briefly re-iterate. The first solution is comprised of exploiting the circuit's structural knowledge, if available. Alternatively, when such information is not known, we can compute tractable bounds using the Cauchy-Schwarz inequality. These two solutions are now presented in more detail in the following text. Finally, recall that the third solution consists of "augmenting" the structure of the circuit. Here, node copies for the shared nodes are used, as discussed in paragraph **c)** of Section 3.2.3 of the main paper.

### A.2.4 Exploiting Structural Knowledge for Exact Covariance Computation

**Tree-structured PCs:** To simplify Eq. (34) we can directly exploit structural knowledge of the DAG. A simple solution could be provided when product nodes $\mathsf{P}^A$ and $\mathsf{P}^B$ are not common ancestors of any node, resulting in the independence $\mathsf{P}^A \perp\!\!\!\perp \mathsf{P}^B$ holding and thus $\mathrm{Cov}[\mathsf{P}^A, \mathsf{P}^B] = 0$. Such a constraint is always given in tree-structured PCs. Notably, the most common structure learner for SPNs such as LearnSPN [Gens and Domingos, 2013], ID-SPN [Rooshenas and Lowd, 2014], and SVD-SPN [Adel et al., 2015] pursue the simplicity bias and induce tree structures.

**Random and Tensorized Binary Tree Structures:** For random and tensorized (RAT) binary tree structures [Peharz et al., 2020], we need to account for the existence of cross-products, to compute the variance of product nodes. For binary tree structures of this kind, we can thus leverage the characteristic that product nodes factorize two completely independent partitions. In the context of this particular structure with binary partitioning, the variance for a RAT sum node, with its children having their scope on graph partitions $L$ and $R$, can be rewritten in the following way:

$$\begin{aligned}\mathrm{Var}[\mathsf{S}] &= \mathrm{Var}\left[\sum_l \sum_r w_{l,r}\delta_{l,r}\mathsf{P}_{l,r}\right] &(37)\\ &= \sum_l \sum_r w_{l,r}^2 \mathrm{Var}[\delta_{l,r}\mathsf{P}_{l,r}] + \sum_{l,r} \sum_{(l',r')\neq(l,r)} w_{l,r}w_{l',r'}\mathrm{Cov}[\mathsf{P}_{l,r},\mathsf{P}_{l',r'}]\ . &(38)\end{aligned}$$

Consequently, the covariance term between two product nodes in Eq. (38) can be simplified to:

$$\begin{aligned}\mathrm{Cov}[\mathsf{P}_{l,r},\mathsf{P}_{l',r'}] &= \mathbb{E}[\mathsf{P}_{l,r}\mathsf{P}_{l',r'}] - \mathbb{E}[\mathsf{P}_{l,r}]\,\mathbb{E}[\mathsf{P}_{l',r'}] &(39)\\ &= \mathbb{E}\big[\mathsf{S}_l^L\mathsf{S}_r^R\mathsf{S}_{l'}^L\mathsf{S}_{r'}^R\big] - \mathbb{E}\big[\mathsf{S}_l^L\mathsf{S}_r^R\big]\,\mathbb{E}\big[\mathsf{S}_{l'}^L\mathsf{S}_{r'}^R\big]\ . &(40)\end{aligned}$$

Since nodes $\mathsf{S}^L$ and $\mathsf{S}^R$ are from two different partitions, they are independent, and the expectation terms can be separated:

$$\begin{aligned}\mathrm{Cov}[\mathsf{P}_{l,r},\mathsf{P}_{l',r'}] =&\mathbb{E}\big[\mathsf{S}_l^L\mathsf{S}_{l'}^L\big]\,\mathbb{E}\big[\mathsf{S}_r^R\mathsf{S}_{r'}^R\big] - \mathbb{E}\big[\mathsf{S}_l^L\big]\,\mathbb{E}\big[\mathsf{S}_r^R\big]\,\mathbb{E}\big[\mathsf{S}_{r'}^R\big]\,\mathbb{E}\big[\mathsf{S}_{l'}^L\big] &(41)\\ =&\big(\mathbb{E}\big[\mathsf{S}_l^L\big]\,\mathbb{E}\big[\mathsf{S}_{l'}^L\big] + \mathrm{Cov}\big[\mathsf{S}_l^L,\mathsf{S}_{l'}^L\big]\big)\big(\mathbb{E}\big[\mathsf{S}_r^R\big]\,\mathbb{E}\big[\mathsf{S}_{r'}^R\big] + \mathrm{Cov}\big[\mathsf{S}_r^R,\mathsf{S}_{r'}^R\big]\big)\\ &- \mathbb{E}\big[\mathsf{S}_l^L\big]\,\mathbb{E}\big[\mathsf{S}_r^R\big]\,\mathbb{E}\big[\mathsf{S}_{l'}^L\big]\,\mathbb{E}\big[\mathsf{S}_{r'}^R\big] &(42)\\ =&\mathbb{E}\big[\mathsf{S}_l^L\big]\,\mathbb{E}\big[\mathsf{S}_r^R\big]\,\mathbb{E}\big[\mathsf{S}_{l'}^L\big]\,\mathbb{E}\big[\mathsf{S}_{r'}^R\big]\\ &+ \mathrm{Cov}\big[\mathsf{S}_l^L,\mathsf{S}_{l'}^L\big]\,\mathbb{E}\big[\mathsf{S}_r^R\big]\,\mathbb{E}\big[\mathsf{S}_{r'}^R\big]\\ &+ \mathrm{Cov}\big[\mathsf{S}_r^R,\mathsf{S}_{r'}^R\big]\,\mathbb{E}\big[\mathsf{S}_l^L\big]\,\mathbb{E}\big[\mathsf{S}_{l'}^L\big]\\ &+ \mathrm{Cov}\big[\mathsf{S}_l^L,\mathsf{S}_{l'}^L\big]\,\mathrm{Cov}\big[\mathsf{S}_r^R,\mathsf{S}_{r'}^R\big]\\ &- \mathbb{E}\big[\mathsf{S}_l^L\big]\,\mathbb{E}\big[\mathsf{S}_r^R\big]\,\mathbb{E}\big[\mathsf{S}_{l'}^L\big]\,\mathbb{E}\big[\mathsf{S}_{r'}^R\big] &(43)\\ =&\mathrm{Cov}\big[\mathsf{S}_l^L,\mathsf{S}_{l'}^L\big]\,\mathbb{E}\big[\mathsf{S}_r^R\big]\,\mathbb{E}\big[\mathsf{S}_{r'}^R\big]\\ &+ \mathrm{Cov}\big[\mathsf{S}_r^R,\mathsf{S}_{r'}^R\big]\,\mathbb{E}\big[\mathsf{S}_l^L\big]\,\mathbb{E}\big[\mathsf{S}_{l'}^L\big]\\ &+ \mathrm{Cov}\big[\mathsf{S}_l^L,\mathsf{S}_{l'}^L\big]\,\mathrm{Cov}\big[\mathsf{S}_r^R,\mathsf{S}_{r'}^R\big]\ . &(44)\end{aligned}$$

We can see that the covariance of two product nodes now depends on the covariance of the input sum nodes of the same partition (i.e. $L$ or $R$), for which we can plug in Eq. (22).

### A.2.5 It's Somewhere in Here – Covariance Bounds

As previously described, knowledge about the specific PC structure can facilitate the covariance computation that otherwise could result in a combinatorial explosion. When such knowledge is not available, we can alternatively obtain a lower and upper bound of the covariance in a tractable way, making use of the Cauchy-Schwarz inequality:

$$\begin{aligned}&\mathrm{Cov}[\mathsf{N}_i,\mathsf{N}_j]^2 \leq \mathrm{Var}[\mathsf{N}_i]\,\mathrm{Var}[\mathsf{N}_j] &(45)\\ \Leftrightarrow\quad &\mathrm{Cov}[\mathsf{N}_i,\mathsf{N}_j]\ \in \left[-\sqrt{\mathrm{Var}[\mathsf{N}_i]\,\mathrm{Var}[\mathsf{N}_j]}, +\sqrt{\mathrm{Var}[\mathsf{N}_i]\,\mathrm{Var}[\mathsf{N}_j]}\right]\ . &(46)\end{aligned}$$

### A.2.6 Leaf Nodes

Finally, as the leaf nodes are free of any dropout Bernoulli variables, their expectation, variance, and covariance degrade to the leaf node value and zero respectively, i.e.:

$$\begin{aligned}\mathbb{E}[\mathsf{L}] &= \mathsf{L}\,, &(47)\\ \mathrm{Var}[\mathsf{L}] &= 0\,, &(48)\\ \mathrm{Cov}[\mathsf{L}_i,\mathsf{L}_j] &= 0\,. &(49)\end{aligned}$$

### A.2.7 Classification Uncertainty

For classification in PCs, for a given sample $\mathbf{x} \sim \mathbf{X}$, it is common to obtain the data conditional class confidence $p(y_i|\mathbf{x})$ by using Bayes' rule with the class conditional root nodes $p(\mathbf{x}|y_i)$ and the class priors $p(y_i) = c_i$. That is, every class $y_i$ has a corresponding root node $\mathsf{S}_i$, representing $p(\mathbf{x}|y_i)$. Following the earlier sections' elaborations, we can then similarly derive the variance as an uncertainty proxy in a classification context. Making use of Bayes' rule, we obtain the posterior for class $y_i$ as follows:

$$p(y_i|\mathbf{x}) = \frac{p(\mathbf{x}|y_i)p(y_i)}{\sum_j p(\mathbf{x}|y_j)p(y_j)} = \frac{\mathsf{S}_i c_i}{\sum_j \mathsf{S}_j c_j} \,. \tag{50}$$

Following existing notation, we will abbreviate the $i$-th root node with $\mathsf{S}_i$ and the $i$-th class prior with $c_i$ for simplicity.

The expectation and variance of the posterior are that of a random variable ratio, $\mathbb{E}\left[\frac{A}{B}\right]$ and $\text{Var}\left[\frac{A}{B}\right]$, with $A = \mathsf{S}_i c_i$ and $B = \sum_j \mathsf{S}_j c_j$. This ratio is generally not well-defined, but can be approximated with a second-order Taylor approximation [Seltman, 2018]:

$$\mathbb{E}\left[\frac{A}{B}\right] \approx \frac{\mathbb{E}[A]}{\mathbb{E}[B]} - \frac{\text{Cov}[A,B]}{(\mathbb{E}[B])^2} + \frac{\text{Var}[B]\,\mathbb{E}[A]}{(\mathbb{E}[B])^3} \,, \tag{51}$$

$$\text{Var}\left[\frac{A}{B}\right] \approx \frac{\mathbb{E}[A]^2}{\mathbb{E}[B]^2}\left[\frac{\text{Var}[A]}{\mathbb{E}[A]^2} - 2\frac{\text{Cov}[A,B]}{\mathbb{E}[A]\,\mathbb{E}[B]} + \frac{\text{Var}[B]}{\mathbb{E}[B]^2}\right] . \tag{52}$$

We will now resolve every component of Eqs. (51) and (52). The expectations can be expressed directly as:

$$\mathbb{E}[A] = \mathbb{E}[\mathsf{S}_i c_i] = \mathbb{E}[\mathsf{S}i]\, c_i \,, \tag{53}$$

$$\mathbb{E}[B] = \mathbb{E}\left[\sum_j \mathsf{S}_j c_j\right] = \sum_j \mathbb{E}[\mathsf{S}_j]\, c_j \,. \tag{54}$$

For the variances we obtain:

$$\text{Var}[A] = \text{Var}[\mathsf{S}_i c_i] = \text{Var}[\mathsf{S}_i]\, c_i^2 \,, \tag{55}$$

$$\text{Var}[B] = \text{Var}\left[\sum_j \mathsf{S}_j c_j\right] \tag{56}$$

$$= \sum_j \text{Var}[\mathsf{S}_j]\, c_j^2 + \sum_{j_1 \neq j_2} \text{Cov}[\mathsf{S}_{j_1}, \mathsf{S}_{j_2}]\, c_{j_1} c_{j_2} \,. \tag{57}$$

Finally, following Eq. (27), the covariance term between a root node and the sum of all root nodes can be decomposed as follows:

$$\mathrm{Cov}[A,B] = \mathrm{Cov}\left[\mathsf{S}_i c_i, \sum_j \mathsf{S}_j c_j\right] \tag{58}$$

$$= \mathbb{E}\left[\mathsf{S}_i c_i \cdot \sum_j \mathsf{S}_j c_j\right] - \mathbb{E}[\mathsf{S}_i c_i]\,\mathbb{E}\left[\sum_j \mathsf{S}_j c_j\right] \tag{59}$$

$$= \mathbb{E}\left[\sum_j \mathsf{S}_j \mathsf{S}_i c_i c_j\right] - \mathbb{E}[\mathsf{S}_i c_i]\,\mathbb{E}\left[\sum_j \mathsf{S}_j c_j\right] \tag{60}$$

$$= c_i \sum_j c_j \mathbb{E}[\mathsf{S}_i \mathsf{S}_j] - c_i \mathbb{E}[\mathsf{S}_i] \sum_j c_j \mathbb{E}[\mathsf{S}_j] \tag{61}$$

$$= c_i \sum_j c_j \left(\mathrm{Cov}[\mathsf{S}_i, \mathsf{S}_j] + \mathbb{E}[\mathsf{S}_i]\,\mathbb{E}[\mathsf{S}_j]\right) - c_i \mathbb{E}[\mathsf{S}_i] \sum_j c_j \mathbb{E}[\mathsf{S}_j] \tag{62}$$

$$= c_i \left(\left(\sum_j c_j \mathrm{Cov}[\mathsf{S}_i, \mathsf{S}_j]\right) + \left(\mathbb{E}[\mathsf{S}_i] \sum_j c_j \mathbb{E}[\mathsf{S}_j]\right) - \left(\mathbb{E}[\mathsf{S}_i] \sum_j c_j \mathbb{E}[\mathsf{S}_j]\right)\right) \tag{63}$$

$$= c_i \sum_j c_j \mathrm{Cov}[\mathsf{S}_i, \mathsf{S}_j] \quad . \tag{64}$$

As previously described but repeated for emphasis, the last term $\mathrm{Cov}[\mathsf{S}_i, \mathsf{S}_j]$ can be resolved with one of the three methods presented in Section 3.2.3 of the main paper, i.e., by exploiting structural knowledge (see also Appendix A.2.4), or by computing bounds with Eq. (45) of Appendix A.2.5, or by "augmenting" the structure with node copies of shared nodes.

While Eq. (52) is seemingly simple, this particular formulation implies statistical independence between $A$ and $B$. Since $B$ is a sum over all $A$, this independence naturally does not hold. Therefore, the solution given here is only an approximation of the true second-order Taylor approximation. In the following we extend the formulation of Seltman [2018] and take into account the dependencies between root nodes $\mathsf{S}_i$ and their sum $\sum_i \mathsf{S}_i$. For simplicity, we express Eq. (50) as a function $f$ of variables $\mathsf{S}_1, \dots, \mathsf{S}_C$.

$$f(\mathsf{S}_1, \dots, \mathsf{S}_C) = f(\boldsymbol{\theta}) + \sum_i \frac{\partial}{\partial \mathsf{S}_i} f(\boldsymbol{\theta})\,(\mathsf{S}_i - \theta_i) + \sum_i \sum_j \frac{\partial^2}{\partial \mathsf{S}_i \partial \mathsf{S}_j} f(\boldsymbol{\theta})\,(\mathsf{S}_i - \theta_i)\,(\mathsf{S}_j - \theta_j) + R \quad , \tag{65}$$

where $R$ is the remainder of smaller orders. To abbreviate equations, we introduce $Z = \sum_j \mathsf{S}_j c_j$ and $Z_{\setminus i} = \sum_{j \neq i} \mathsf{S}_j c_j$. To resolve the first-order term, we need the first partial derivative at $\mathsf{S}_i$.

$$\frac{\partial}{\partial \mathsf{S}_i} f(\mathsf{S}_1, \dots, \mathsf{S}_C) = \frac{Z \frac{\partial \mathsf{S}_i c_i}{\partial \mathsf{S}_i} - \mathsf{S}_i c_i \frac{\partial}{\partial \mathsf{S}_i} Z}{Z^2} \tag{66}$$

$$= \frac{c_i Z - \mathsf{S}_i c_i^2}{Z^2} \tag{67}$$

$$= c_i \frac{Z - \mathsf{S}_i c_i}{Z^2} \tag{68}$$

$$= c_i \frac{Z_{\setminus i}}{Z^2} \quad . \tag{69}$$

We continue with the second partial derivative at $S_k$ and make a distinction for $k = i$ and $k \neq i$. For the second partial derivative at $S_i$ we get

$$\frac{\partial^2}{\partial \mathsf{S}_i \partial \mathsf{S}_i} f(\mathsf{S}_1, \dots, \mathsf{S}_C) = \frac{Z^2 \left(\frac{\partial}{\partial \mathsf{S}_i} Z_{\setminus i}\right) - Z_{\setminus i} \left(\frac{\partial}{\partial \mathsf{S}_i} Z^2\right)}{Z^4} \tag{70}$$

$$= \frac{-2 c_i Z_{\setminus i} Z}{Z^4} \tag{71}$$

$$= -2 c_i \frac{Z_{\setminus i}}{Z^3} \quad . \tag{72}$$

For the second partial derivative at $S_k$ with $k \neq i$ we get

$$\begin{aligned}
\frac{\partial^2}{\partial \mathsf{S}_i \partial \mathsf{S}_k} f(\mathsf{S}_1, \dots, \mathsf{S}_C) &= \frac{Z^2 \left(\frac{\partial}{\partial \mathsf{S}_k} Z_{\setminus i}\right) - Z_{\setminus i} \left(\frac{\partial}{\partial \mathsf{S}_k} Z^2\right)}{Z^4} && (73)\\
&= \frac{c_k Z^2 - 2 c_k Z Z_{\setminus i}}{Z^4} && (74)\\
&= \frac{c_k Z - 2 c_k Z_{\setminus i}}{Z^3} && (75)\\
&= c_k \frac{Z - 2 Z_{\setminus i}}{Z^3} && (76)\\
&= c_k \frac{Z - 2\left(Z - \mathsf{S}_i c_i\right)}{Z^3} && (77)\\
&= c_k \frac{-Z - 2\mathsf{S}_i c_i}{Z^3} && (78)\\
&= -c_k \frac{Z + 2\mathsf{S}_i c_i}{Z^3} \quad . && (79)
\end{aligned}$$

Using Eqs. (69), (72) and (79), and $\theta_i = \mathbb{E}[\mathsf{S}_i]$ we obtain

$$\begin{aligned}
f(\mathsf{S}_1, \dots, \mathsf{S}_C) = f(\mathbb{E}[\mathsf{S}_1], \dots, \mathbb{E}[\mathsf{S}_C]) + \sum_i \left(\frac{c_i \mathbb{E}[Z_{\setminus i}]}{\mathbb{E}[Z]^2}\right) (\mathsf{S}_i - \mathbb{E}[\mathsf{S}_i]) \\
+ \sum_i \left( \sum_{j \neq i} \left( -c_j \frac{\mathbb{E}[Z] + 2\mathbb{E}[\mathsf{S}_i]\, c_i}{\mathbb{E}[Z]^3} (\mathsf{S}_i - \mathbb{E}[\mathsf{S}_i]) (\mathsf{S}_j - \mathbb{E}[\mathsf{S}_j]) \right) \right. \\
\left. - 2c_i \frac{\mathbb{E}[Z_{\setminus i}]}{\mathbb{E}[Z]^3} (\mathsf{S}_i - \mathbb{E}[\mathsf{S}_i])^2 \right) + R \quad . \qquad (80)
\end{aligned}$$

Now that we have derived the second order Taylor approximation of Eq. (50), we need to take the expectation and variance of $f$. We begin with the expectation as follows

$$\begin{aligned}
\mathbb{E}[f(\mathsf{S}_1, \dots, \mathsf{S}_C)] = f(\mathbb{E}[\mathsf{S}_1], \dots, \mathbb{E}[\mathsf{S}_C]) + \sum_i \left(\frac{c_i \mathbb{E}[Z]}{\mathbb{E}[Z]^2}\right) \overbrace{\mathbb{E}[\mathsf{S}_i - \mathbb{E}[\mathsf{S}_i]]}^{=\mathbb{E}[\mathsf{S}_i] - \mathbb{E}[\mathsf{S}_i] = 0} \\
+ \sum_i \left( \sum_{j \neq i} \left( -c_j \frac{\mathbb{E}[Z] + 2\mathbb{E}[\mathsf{S}_i]\, c_i}{\mathbb{E}[Z]^3} \overbrace{\mathbb{E}[(\mathsf{S}_i - \mathbb{E}[\mathsf{S}_i]) (\mathsf{S}_j - \mathbb{E}[\mathsf{S}_j])]}^{=\mathrm{Cov}[\mathsf{S}_i, \mathsf{S}_j]} \right) \right. \\
\left. - 2c_i \frac{\mathbb{E}[Z_{\setminus i}]}{\mathbb{E}[Z]^3} \overbrace{\mathbb{E}\left[(\mathsf{S}_i - \mathbb{E}[\mathsf{S}_i])^2\right]}^{=\mathrm{Var}[\mathsf{S}_i]} \right) \quad , \qquad (81)
\end{aligned}$$

which simplifies to

$$\mathbb{E}[f(\mathsf{S}_1, \dots, \mathsf{S}_C)] = f(\mathbb{E}[\mathsf{S}_1], \dots, \mathbb{E}[\mathsf{S}_C]) + \sum_i \left( \sum_{j \neq i} \left( -c_j \frac{\mathbb{E}[Z] + 2\mathbb{E}[\mathsf{S}_i]\, c_i}{\mathbb{E}[Z]^3} \mathrm{Cov}[\mathsf{S}_i, \mathsf{S}_j] \right) - 2c_i \frac{\mathbb{E}[Z_{\setminus i}]}{\mathbb{E}[Z]^3} \mathrm{Var}[\mathsf{S}_i] \right) . \qquad (82)$$

We continue with the variance of $f$

$$\begin{aligned}\operatorname{Var}[f(\mathsf{S}_1,\ldots,\mathsf{S}_C)] = \operatorname{Var}[f(\mathbb{E}[\mathsf{S}_1],\ldots,\mathbb{E}[\mathsf{S}_C])] + \sum_i \operatorname{Var}\left[c_i \frac{\mathbb{E}[Z_{\setminus i}]}{\mathbb{E}[Z]^2}(\mathsf{S}_i - \mathbb{E}[\mathsf{S}_i])\right] \\ + \sum_i \left( \sum_{j\neq i} \left(-c_j \frac{2\mathbb{E}[\mathsf{S}_i]\, c_i + \mathbb{E}[Z]}{\mathbb{E}[Z]^3}\right)^2 \operatorname{Var}[(\mathsf{S}_i - \mathbb{E}[\mathsf{S}_i])(\mathsf{S}_j - \mathbb{E}[\mathsf{S}_j])] \right. \\ \left. + \left(-2c_i \frac{\mathbb{E}[Z_{\setminus i}]}{\mathbb{E}[Z]^3}\right) \operatorname{Var}\left[(\mathsf{S}_i - \mathbb{E}[\mathsf{S}_i])^2\right]\right) \quad . \end{aligned} \tag{83}$$

Resolving each variance term simplifies the equation to

$$\begin{aligned}\operatorname{Var}[f(\mathsf{S}_1,\ldots,\mathsf{S}_C)] = \sum_i \left(c_i \frac{\mathbb{E}[Z_{\setminus i}]}{\mathbb{E}[Z]^2}\right)^2 \operatorname{Var}[\mathsf{S}_i] \\ + \sum_i \left( \sum_{j\neq i} \left(-c_j \frac{2\mathbb{E}[\mathsf{S}_i]\, c_i + \mathbb{E}[Z]}{\mathbb{E}[Z]^3}\right)^2 \left(\operatorname{Var}[\mathsf{S}_i\mathsf{S}_j] - \mathbb{E}[\mathsf{S}_i]^2 \operatorname{Var}[\mathsf{S}_j] - \mathbb{E}[\mathsf{S}_j]^2 \operatorname{Var}[\mathsf{S}_i]\right) \right. \\ \left. + \left(-2c_i \frac{\mathbb{E}[Z_{\setminus i}]}{\mathbb{E}[Z]^3}\right) \left(\operatorname{Var}\left[\mathsf{S}_i^2\right] - 4\mathbb{E}[\mathsf{S}_i]^2 \operatorname{Var}[\mathsf{S}_i]\right)\right) \quad . \end{aligned} \tag{84}$$

We now end up with a variance term that needs further resolution: the product of two sum nodes.

$$\operatorname{Var}[\mathsf{S}_i\mathsf{S}_j] = \operatorname{Var}\left[\sum_{k_i} w_{k_i}^{\mathsf{S}_i} \mathsf{N}_{k_i}^{\mathsf{S}_i} \sum_{k_j} w_{k_i}^{\mathsf{S}_i} \mathsf{N}_{k_i}^{\mathsf{S}_i}\right] \tag{85}$$

$$= \operatorname{Var}\left[w_1^{\mathsf{S}_i} w_1^{\mathsf{S}_j} \mathsf{N}_1^{\mathsf{S}_i} \mathsf{N}_1^{\mathsf{S}_j} + \cdots + w_1^{\mathsf{S}_i} w_{|\mathsf{S}_j|}^{\mathsf{S}_j} \mathsf{N}_1^{\mathsf{S}_i} \mathsf{N}_{|\mathsf{S}_j|}^{\mathsf{S}_j} \cdots + w_{|\mathsf{S}_i|}^{\mathsf{S}_i} w_{|\mathsf{S}_j|}^{\mathsf{S}_j} \mathsf{N}_{|\mathsf{S}_i|}^{\mathsf{S}_i} \mathsf{N}_{|\mathsf{S}_j|}^{\mathsf{S}_j}\right] \tag{86}$$

$$= \sum_{k_i} \sum_{k_j} \operatorname{Var}\left[w_{k_i}^{\mathsf{S}_i} w_{k_j}^{\mathsf{S}_j} \mathsf{N}_{k_i}^{\mathsf{S}_i} \mathsf{N}_{k_j}^{\mathsf{S}_j}\right] \tag{87}$$

$$= \sum_{k_i} \sum_{k_j} \left(w_{k_i}^{\mathsf{S}_i} w_{k_j}^{\mathsf{S}_j}\right)^2 \operatorname{Var}\left[\mathsf{N}_{k_i}^{\mathsf{S}_i} \mathsf{N}_{k_j}^{\mathsf{S}_j}\right] \quad . \tag{88}$$

Analogous to the discussions in Appendix A.2.3, the complexity of computing the variance of the product of two product nodes can be addressed through the incorporation of supplementary structural knowledge (refer to Appendix A.2.4 for further details) or by utilizing crude approximations that induce local independence. An example of such an approximation is expressed as $\operatorname{Var}\left[\mathsf{N}_{k_i}^{\mathsf{S}_i} \mathsf{N}_{k_j}^{\mathsf{S}_j}\right] = \operatorname{Var}\left[\mathsf{N}_{k_i}^{\mathsf{S}_i}\right] \operatorname{Var}\left[\mathsf{N}_{k_j}^{\mathsf{S}_j}\right]$. With Eqs. (82), (84) and (85) we have now reduced the second-order Taylor posterior approximation to quantities that we are able to compute from a single bottom up pass through the PC graph.

## B TRACTABLE DROPOUT INFERENCE PSEUDOCODE ALGORITHM

Analogously to the conventional probabilistic inference on PCs, TDI is performed by evaluating the circuit bottom-up. For clarity and completeness, we present the pseudocode of the bottom-up pass with TDI in Algorithm 1. While the bottom-up pass can be implemented in a variety of different ways, we have decided on a recursive version in the presented pseudocode for shortness and clarity. We start by initializing empty maps `E`, `Var`, and `Cov` that map nodes to their expectation, variance, and covariance values. Given a node $\mathsf{N}$, some data sample $\boldsymbol{x}$, and the dropout probability $p$, we then now traverse the graph below $\mathsf{N}$ and require, that the `tdi` procedure has been called on all children, ensuring that the `E`, `Var`, and `Cov` of all $\mathsf{N}_i \in \mathbf{ch}(\mathsf{N})$ are populated. We then continue to compute the covariance between all child nodes of $\mathsf{N}$, followed by the expectation and finally the variance of $\mathsf{N}$. All three procedure calls, `expectation`, `variance`, and `covariance`, refer to the equations given in the main body in Section 3.2 for sum, product, and leaf nodes respectively.

**Algorithm 1** Tractable Dropout Inference: `tdi`$(\mathsf{N}, \boldsymbol{x}, p)$

```
Require: PC root node $\mathsf{N}$; Data sample $\boldsymbol{x}$; Dropout probability $p$; Empty maps E, Var, Cov.
  for all $\mathsf{N}_i \in \mathbf{ch}(\mathsf{N})$ do
      tdi$(\mathsf{N}_i, \boldsymbol{x}, p)$                                ▷ Populate maps for child nodes recursively
  for all $\mathsf{N}_i \in \mathbf{ch}(\mathsf{N})$ do
      for all $\mathsf{N}_j \in \mathbf{ch}(\mathsf{N})$ do
          Cov$[\mathsf{N}_i, \mathsf{N}_j]$ = covariance$(\mathsf{N}_i, \mathsf{N}_j, \boldsymbol{x}, p)$     ▷ See Eqs. (6), (7) and (11)
  E$[\mathsf{N}]$ = expectation$(\mathsf{N}, \boldsymbol{x}, p)$                  ▷ See Eqs. (2), (3) and (11)
  Var$[\mathsf{N}]$ = variance$(\mathsf{N}, \boldsymbol{x}, p)$                    ▷ See Eqs. (4), (5) and (11)
```

## C EXPERIMENTAL SETUP

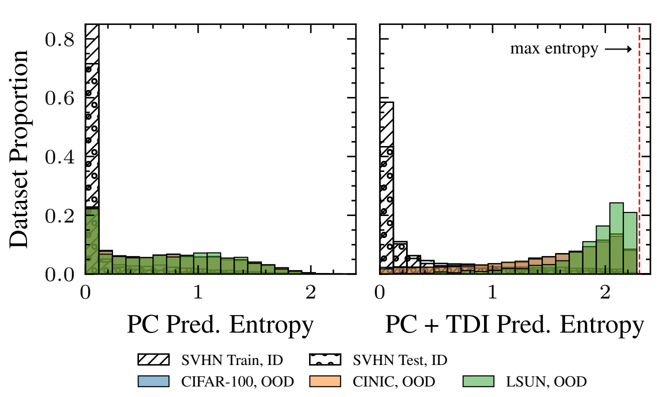
For our experiments, we implemented TDI based on the publicly-available PyTorch implementation of RAT-SPNs[1] in the *SPFlow* software package [Molina et al., 2019]. Our code is available at `https://github.com/ml-research/tractable-dropout-inference`. All our models were trained for 200 epochs with the Adam optimizer [Kingma and Ba, 2015], employing a batch size of 200 and a learning rate of 1e-3. To assess the robustness of probabilistic circuits as probabilistic discriminators, we have run all our experiments with the RAT hyperparameter $\lambda$ set to 1. The latter corresponds to maximizing the class conditional likelihood of a sample computed by the head attributed to the observed label. Regarding the RAT-SPN graph, we make use of the following hyperparameters: $S = 20, I = 20, D = 5, R = 5$ with Gaussian leaves. This configuration returns circuits with 1.65M edges and 1.83M learnable parameters for SVHN (614k are Gaussian leaf parameters), and 1.42M edges and 1.38M parameters for MNIST (157k are Gaussian leaf parameters). We emphasize that all previously described hyperparameter choices correspond to standard, previously considered, choices in the literature. No additional or excessive hyperparameter tuning has been conducted on our side. We refer to the original work of Peharz et al. [2020] for further details on RAT-SPN training and its respective hyperparameters.

For our TDI experiments, we select the dropout parameter $p$ to be $0.1$ for SVHN and $0.2$ for MNIST. This range is adequate to preserve accuracy and simultaneously provide valuable uncertainty estimates. Note how such a choice is consistent and not fundamentally different from dropout probabilities in neural networks (i.e. typically on the same scale in every layer, unless it is the last layer, where sometimes a $p$ of up to 0.5 can be found in the literature). However, we further note that probabilistic circuits are typically sparser than their neural counterparts. A small dropout parameter is thus generally sufficient but nevertheless remains a hyperparameter to be chosen.

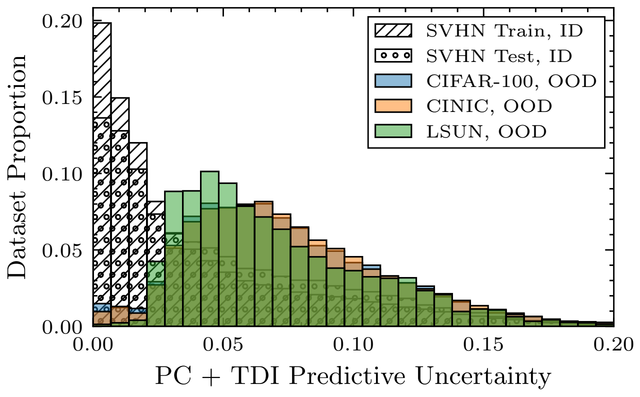
## D PCS WITH TDI CAN DETECT OOD DATA

In the main paper, we have shown that PCs tend to be overconfident on OOD instances while PCs + TDI can overcome this challenge and can detect when an instance comes from a completely different distribution compared to what observed during training. This facilitates the separation between ID and OOD instances with a threshold, a peculiarity that is particularly useful also in practice in many domains. Here, we provide a complementary view of this ability of PCs with TDI. We show the predictive entropy in Fig. 1a and the predictive uncertainty in Fig. 1b obtained on the ID data and on several OOD datasets.

[1] `https://github.com/SPFlow/SPFlow/tree/master/src/spn/experiments/RandomSPNs_layerwise`



(a)



(b)

Figure 1: (a) Histograms for the predictive entropy of PCs and PCs + TDI on ID (shaded) and OOD (colored) datasets in a classification context. PCs (left panel) assign similar, largely indistinguishable predictive entropy to OOD and ID data. In contrast, PCs + TDI (right panel) provides high uncertainty on OOD data, pushing the predictive entropy close to the obtainable maximum and effectively separating them from the known ID dataset. (b) Predictive uncertainty (standard deviation as square root of Eq. (14)) for the predictions provided by PC + TDI on ID (shaded) and OOD data (colored). Complementing the illustration of predictive entropy in (a), the values support the picture that PCs + TDI are more unsure about OOD data and can thus make a distinction.

## E PCS WITH TDI ARE MORE ROBUST TO CORRUPTIONS

We have applied 15 non-trivial natural and synthetic corruptions with respective five levels of severity, as shown in the main body's experimental evidence section. For convenience, we re-iterate that the latter are commonly employed in the literature [Hendrycks and Dietterich, 2019] to test models' robustness. More specifically, corruptions have been used to demonstrate standard neural networks' inability to effectively handle corrupted data. Since we have observed an initial similar inability in PCs, they thus form a sensible test to evaluate the robustness of PCs versus PCs + TDI. To provide some visual intuition, a respective SVHN illustration is shown in Fig. 2.

In our main body, we have shown the performance of PCs and PCs + TDI for 4 such corruptions and thus have provided empirical evidence for the robustness of PCs with TDI and their ability to attribute increased uncertainty to corrupted data. Here, we provide the full set of experiments for all 15 corruptions. Recall that for a model to successfully detect the corruption, it should assign a progressive increase in predictive entropy in accordance with the corruption severity.

The full quantitative results of Fig. 3 empirically affirm our conclusions drawn in the main body: PCs with TDI are generally equally or more robust at various severity levels for all corruptions, generally preserving equal or even higher predictive accuracy in several cases. At the same time, the assigned predictive entropy is substantially larger with an increasing level of corruption with TDI. Finally, we emphasize that in the three cases where PCs with TDI do not provide a large and successively increasing measure of entropy, i.e. zoom blur, pixelation, and jpeg compression, their respective accuracy in Fig. 3 is almost fully preserved. In other words, the model does not provide a large uncertainty because it already provides a correct and robust prediction.

## F SOCIETAL IMPACT

We believe that quantification of model uncertainty, as a direction to contribute to overall robustness, can have a broad positive societal impact. Colloquially speaking, an indication of when to "trust" the model is a valuable tool for users and practitioners, especially in safety-critical applications. That noted, gauging model uncertainty does not absolve users from careful considerations of other potentially harmful effects, particularly, the involvement of various forms of bias through training data, as these are then considered "certain" by definition.

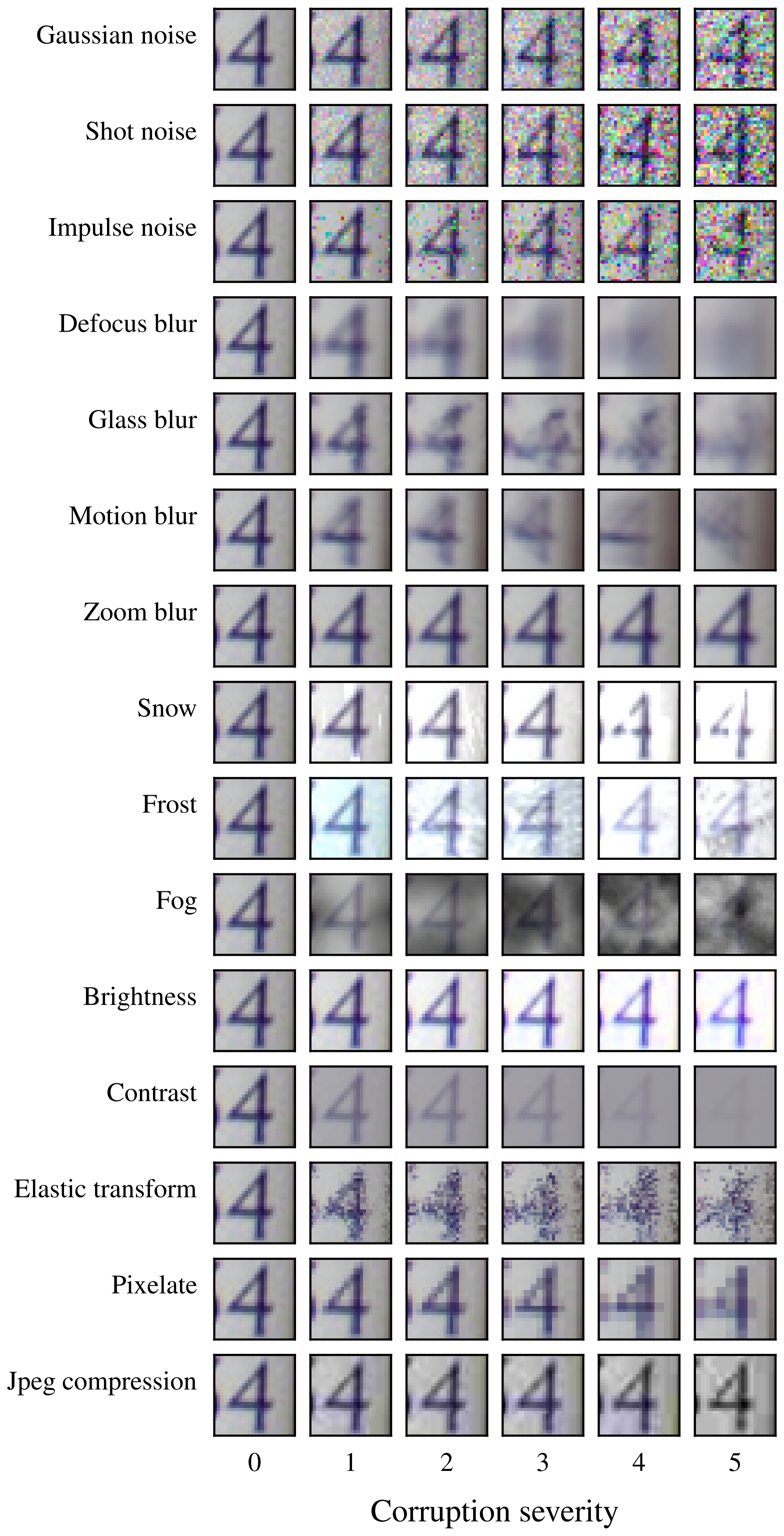


Figure 2: Visual illustration of the 15 different corruptions with five increasing levels of severity as introduced in Hendrycks and Dietterich [2019] on an SVHN test sample.

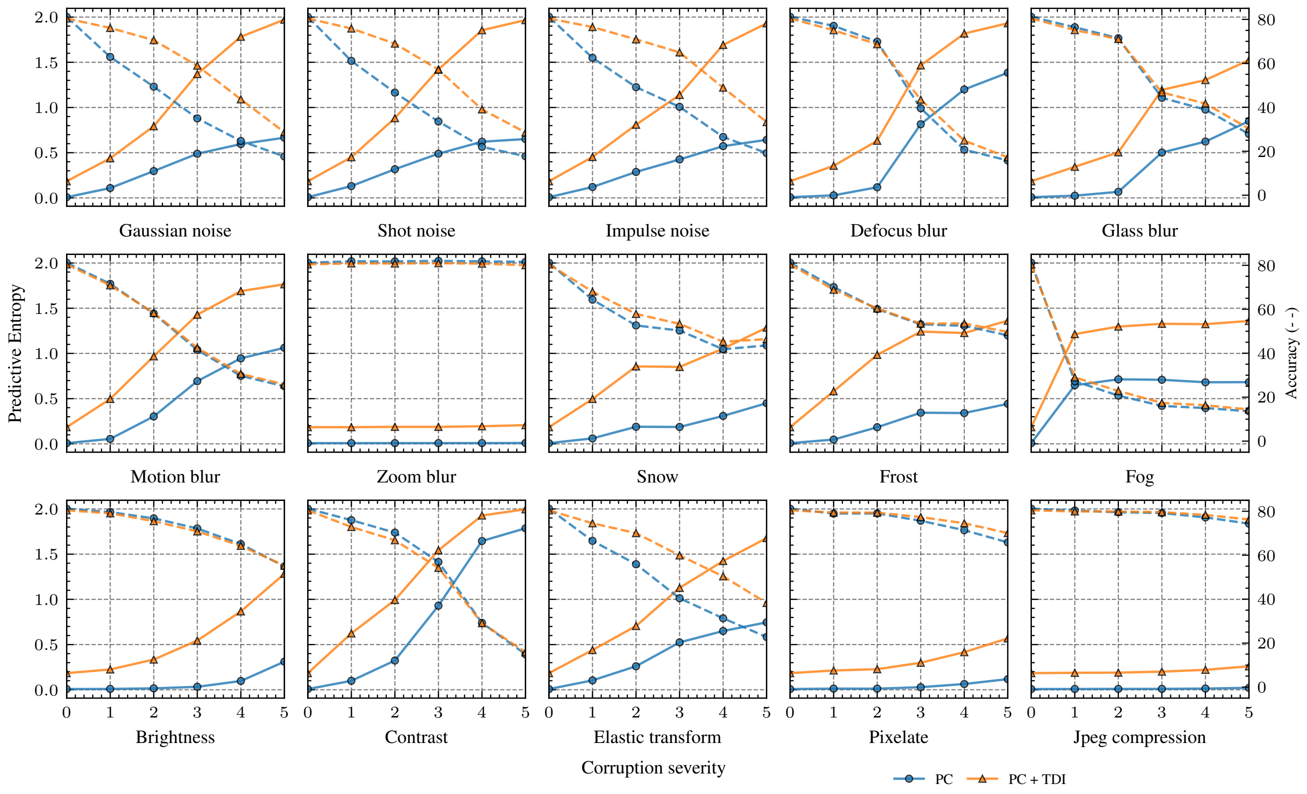


Figure 3: Predictive entropy (left y-axis) and accuracy (right y-axis) of PCs (blue curve, circle markers) and PCs + TDI (orange, triangles) for increasingly corrupted SVHN data. 15 natural and synthetic corruptions are introduced at five severity levels. PCs with TDI can detect the distribution shift by assigning higher predictive entropy with increasing severity, while at the same time being more robust in predictive accuracy against the corruption, compared to PCs. We emphasize that in the three cases where PCs + TDI do not provide a large and successively increasing measure of entropy, i.e. zoom blur, pixelation, and jpeg compression, their respective accuracy (dashed) is almost fully preserved. The model is already correct and certain.

## References

Tameem Adel, David Balduzzi, and Ali Ghodsi. Learning the structure of sum-product networks via an svd-based algorithm. In *UAI*, 2015.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

Robert Gens and Pedro Domingos. Learning the Structure of Sum-Product Networks. In *ICML*, 2013.

Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Pranav Subramani, Nicola Di Mauro, Pascal Poupart, and Kristian Kersting. Spflow: An easy and extensible library for deep probabilistic learning using sum-product networks. *arXiv preprint arXiv:1901.03704*, 2019.

Robert Peharz, Antonio Vergari, Karl Stelzner, Alejandro Molina, Xiaoting Shao, Martin Trapp, Kristian Kersting, and Zoubin Ghahramani. Random sum-product networks: A simple and effective approach to probabilistic deep learning. In *UAI*, 2020.

Amirmohammad Rooshenas and Daniel Lowd. Learning sum-product networks with direct and indirect variable interactions. In *ICML*, 2014.

Howard Seltman. Approximations for mean and variance of a ratio. Technical report, CMU, 2018.