We would like to thank the reviewers for their thoughtful comments and valuable suggestions.

[R1,R3] Using our techniques for density estimation, not classification: our techniques for complexity scaling during learning, numerical precision reduction and for finding the Pareto optimal set of configurations apply directly to the scenario where the goal is density estimation. But as R3 notes, the feature and sensor pruning operation is not applicable if one wants to keep the full joint distribution: for example, the black line in Figure 3(b) shows only model complexity scaling in terms of computation cost. The paper focuses on classification because it addresses restrictions of embedded sensory applications, like in the smartphone-based human activity recognition benchmark. However, having a joint distribution available is still an essential property of ACs/SPNs, and keeps them robust to missing data from unavailable sensors as shown in Figure 3(d) (green and magenta lines).

[R1] Applying this more broadly to Markov networks or probabilistic programs: Our method applies to all probabilistic models that are compute-efficient at prediction time, because such models can be compiled into circuits. Please see the UAI-2019 tutorial slides on Tractable Probabilistic Models by Van den Broeck, Vergari and Di Mauro for examples of how to compile PGMs (see also example below) and probabilistic programs into circuits. In fact, the state of the art for discrete probabilistic program inference is to compile into tractable circuits, where our techniques become directly applicable (see work of Holtzen, Vlasselaer, Riguzzi, etc.). We will clarify this point in the paper.

[R1] Being "restricted to binary random variables": All the datasets in our experiments include multi-valued features: we introduce a binary indicator variable for each of their values. Such a one-hot encoding is standard in SPN learning and deep learning, and is w.l.o.g.; thus, our approach supports categorical variables already. In principle, by using another type of SPN learner, our method could also support continuous random variables, which can be modeled using univariate Gaussian leafs in the circuit. Our algorithms are agnostic to the leaf distributions used.

[R1] About "the method is similar to backward feature selection": There is a lot more going on in our algorithm: precision scaling, model simplification, hardware-aware prediction, cost measurements, etc. In addition, pruning features directly impacts computation and sensory interfacing costs, as noted by R3 and discussed more below.

[R1] Improving the pseudocode: Thanks for this valuable feedback, we will improve the pseudocode as you suggest.

[R1] Including datasets for density estimation: We considered datasets from both the probabilistic models and the general machine learning communities, to demonstrate that our method is effective regardless of the data's nature.

[R2] Runtime overhead: The set of Pareto configurations is always determined off-line, and each resulting model subsequently saved in memory. The processor will fetch and execute them at run-time according to accuracy and cost needs or to comply with a given policy. As such, there is memory overhead but no computational overhead. In addition, predictions must be made at a much higher frequency than configuration changes are necessary in most applications, so there is no need to run the optimization for each inference query. In the on-line deployment experiment of Section 6.1, the total energy-cost overhead is of less than 5% of the prediction cost. We will include this breakdown in the paper.

[R2] Using other performance metrics: Other application-specific performance metrics can replace the accuracy terms in Algorithms 1 and 3 and can be modified as needed (e.g. medical applications often aim to balance precision and recall, and may favor the latter at night under scarce medical supervision). We will clarify this in the paper.

[R3] On the correlation between computation and feature cost: Thank you for highlighting this interesting correlation, which is induced by the AC pruning component of our search strategy. We will comment on this aspect and include an empirical analysis: e.g. for the HAR benchmark, we have seen a reduction on $\mathrm{C_{AC}}$ of 4 to 8 % per feature pruned.

[R3] Extending the related section: Thanks for the valuable feedback, we will modify the related section as suggested.

[R3,R1] Comparison to other approaches: On the left side of the table below, we report the performance of a Tree Augmented Naive Bayes classifier after its compilation to an AC ($\mathrm{Acc_{te}}$,$\mathrm{C_{AC}}$). We will add these results to the paper, as they show how to apply PGMs to our framework; and that PSDDs have a superior performance on most benchmarks.

[R1] Being restricted to binary classification We will include the 6-class HAR experiment reported in the table below, which proves our method works with multi-class problem and that it is comparable to the TAN baseline.

Banknote	HAR	Houses	Jester	Madelone	Nltcs	Wilt	HAR multi-class Pareto / TAN
92.15, 0.5	96.9, 0.5	97.09, 0.1	72.25, 0.2	65.72, 0.3	91.96, 0.2	97.5, 0.6	91.2,1; 91,0.42; 88,0.3 / TAN: 91,0.4

[R2] The heading "64-bits in Table 2 is unclear:" We thank R2 for noting this, we confirm that it refers to the cost computed for that precision, and we will rename the headers to make this clearer.

[R1,R2,R3] Typos, errors and format changes: We thank R1 for identifying the errors of Algorithm 1 and R3 for identifying several typos. We will make the due corrections. We also thank R1 and R2 for pointing out that Fig. 3 is too small, we will assign it due space.