On hardware-aware probabilistic frameworks for resource constrained embedded applications

Laura I. Galindez Olascoaga[∓] **, Wannes Meert**[†]**, Nimish Shah**[∓]**, Guy Van den Broeck**[‡] **and Marian Verhelst**[∓]

[∓] Electrical Engineering Department, KU Leuven , [†] Computer Science Department, KU Leuven
[‡] Computer Science Department, University of California, Los Angeles

Abstract

Edge reasoning attempts to mitigate latency and privacy shortcomings of cloud computing paradigms. However, it introduces additional challenges linked to the devices' resource constraints and the applications' dynamic conditions. To address these challenges, we have proposed a hardware-aware probabilistic framework that optimizes the target machine learning model under actual hardware constraints. This framework relies on tractable probabilistic models, as they facilitate efficient inference, while exhibiting a number of traits relevant to the application range of interest: robustness to missing data, joint prediction capabilities, explainability, and small data needs. In this work, we expand on this framework by introducing a discriminative-generative approach to model learning, which retains the robustness of a generative model under missing data but can potentially improve its discriminative performance. In addition, we demonstrate how the applicability of this framework goes beyond classification tasks, and can be used for density estimation tasks, relevant to applications such as mobile speaker verification.

1 Introduction

Deep Learning (DL) has recently shown great success in a variety of tasks ranging from computer vision to natural language processing. But constraints inherent to embedded applications call for a unique set of attributes that DL approaches are often not equipped with, such as the ability to handle uncertainty and missing data, and the capability of encoding domain knowledge. Probabilistic models are capable of addressing many of these challenges and can therefore constitute a complementary approach to DL. Yet, their deployment has not been as widespread since traditional exact inference techniques on them can be inefficient or even intractable. Current efforts in the field of Tractable Probabilistic Modeling have been making great strides towards balancing the trade-off between model expressiveness and inference efficiency, while achieving state of the art performance on par with DL approaches [13]. However, these efficiency notions are often given in abstract terms such as *time* and *space* and disregard implementation nuances of the target applications and hardware devices. To address such application-driven limitations, recent works have proposed to infuse probabilistic reasoning frameworks with *hardware-awareness* [3, 4]. Such paradigms enable the exploitation of scalable system properties (such as sensor quality, the number of extracted features, the model complexity, or the precision used for computations) to smartly balance the trade-off between task performance and device resource availability. In particular, these works have focused on the trade-off between accuracy of a classification task and total system-level energy cost of the involved embedded device. This paper builds upon the work in [4] and adds two contributions: 1) a discriminative approach to the model-learning step that capitalizes on the model's ability to encode domain knowledge; and 2) how the framework can be deployed with alternative tasks and performance metrics.

5thWorkshop on Energy Efficient Machine Learning and Cognitive Computing (EMC^2 at NeurIPS 2019).

2 Background

Variables are denoted by upper case letters X and their instantiations by lower case letters x. Sets of variables are denoted in bold upper case $\mathbf{X}$ and their joint instantiations in bold lower case $\mathbf{x}$. The tractable model representation used in this work is the Arithmetic Circuit[1] (AC) [2], a directed acyclic graph where inner nodes represent addition (logical OR gate) or multiplication (logical AND gate) and leaf nodes are real valued. When ACs are used to represent joint probability distributions over a set of random variables $\mathbf{X}$, their leaves encode binary evidence-indicator variables $_{X=x}$ or probabilistic parameters . As such, ACs already encode the formula required to perform inference, and are therefore a well-suited representation for tractable learning. Given an instantiation $\mathbf{f}$ of $\mathbf{F}$ $\mathbf{X}$, the marginal probability $\Pr(\mathbf{f})$ can be computed by setting the indicator variables to 1 if they correspond to instantiations consistent with the observed values, $_{X=x}$ $1_{x\sim\mathbf{f}}$, and subsequently performing an upward pass on the AC. In a binary classification task, one defines a class variable C, a feature set $\mathbf{F}$ and a classification threshold $T = 0.5$, and selects the class C_T for which the condition $\Pr(C|\mathbf{f})$ T is met, where $\Pr(C|\mathbf{f}) = \Pr(C,\mathbf{f})/\Pr(\mathbf{f})$ is calculated wih two upward passes in the AC.

The type of AC used in this work is the Probabilistic Sentential Decision Diagram (PSDD) [6]. The inner nodes of a PSDD alternate between AND gates with two inputs and OR gates with arbitrary number of inputs. A *decision* node is the combination of an OR gate with its AND gate inputs, where the left input of the AND gate is referred to as *prime* (p), and the right referred to as *sub* (s). PSDDs possess a number of syntactic restrictions: 1) Each AND node must be *decomposable*, meaning that their input variables must be disjoint. This property is enforced by a variable tree (*vtree*), a binary tree whose leaves are the random variables (see Fig.3). In each internal node, variables appearing in the left subtree $\mathbf{X}$ are the primes and the ones appearing in the right subtree $\mathbf{Y}$ are the subs. 2) Each *decision* node must be deterministic, meaning that only one of its inputs can be true. Thus, a decision node encodes a joint distribution over disjoint variables $\mathbf{X}$ and $\mathbf{Y}$ given by $\Pr(\mathbf{X},\mathbf{Y}) = \sum_i {}_i \Pr(\mathbf{X})\Pr(\mathbf{Y})$, where the parameter $_i$ is a maximum-likelihood estimate.

3 Hardware-Aware probabilistic framework

Fig. 1 depicts the building blocks of the embedded system considered in this work: 1) sensor interface block, marked in green, where the incoming signals are processed for feature generation; and 2) a computational block, marked in blue, where the AC ( ) computes marginal probabilities.

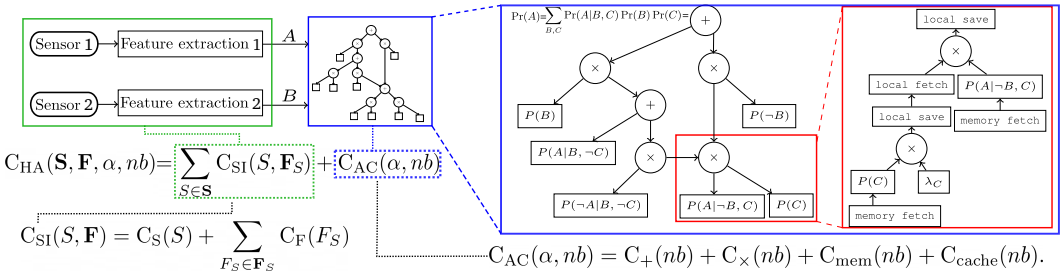

Figure 1: Block diagram of an embedded sensing system and the definition of hardware-aware cost.

3.1 Hardware-aware cost

The *hardware-aware cost* $\mathrm{C_{HA}}$ introduced in [4] describes the total system's hardware-resource consumption during real time data collection and probabilistic inference, and can be expressed in terms of a measurable property, such as energy consumption, throughput or area. Fig. 1 breaks the $\mathrm{C_{HA}}$ down when considering relative energy consumption: sensor interfacing costs $\mathrm{C_{SI}}$ are a function of the sensor and feature sets $\mathbf{S}$; $\mathbf{F}$, and inference-computation costs $\mathrm{C_{AC}}$ are a function of elementary operation costs (multiplication, addition and memory exchanges); and precision in number of bits nb.

[1] An alternative representation of ACs are Sum-Product Networks (SPNs) [9].

3.2 Trade-off search

The trade-off between hardware-aware cost and task-performance is thus determined by four system properties: inference model complexity, the type and number of sensors and features (F, S) and the number of bits used for computations (nb). The goal is to find the system configuration that incurs the lowest cost for a desired inference quality, or the best inference quality for a given hardware cost constraint. In other words, the pareto-optimal set of system configurations is: $\{f_i, F_i, S_i, nb_i\}_{i=1:p}$. The top of Fig. 2 illustrates the proposed technique, which takes places over two main stages: 1) the trade-off space is first searched by tuning scalable system properties; 2) the Pareto-optimal configuration set is then extracted from the search space. This search takes places over several consecutive stages determined by the desired task (classification or density estimation), the type of input to the model learning strategy (data or a probabilistic model), and the capabilities of the hardware (whether e.g. it supports low precision arithmetic or not).

Figure 2: Hardware-aware probabilistic framework and its algorithms.

Model scaling The first step consists of learning a set of models of increasing complexity by either compiling them from a Probabilistic Graphical Model (PGM) or a Probabilistic Program (PP) [2, 5], or by learning their structure [7, 11]. The work in [4] focuses on a classification task, performed on a set of increasingly complex PSDDs. These models are generated by the PSDDlearn algorithm from [7], which incrementally improves the structure of an existing PSDD to better fit the data (by maximizing model log-likelihood). A generative learner was selected since it guarantees robustness against missing data, which in our framework is a natural consequence of feature and sensor pruning. However, generative learning disregards the correlation between the class variable C and the feature variables F. For the framework in [4], the learning algorithm is driven by a vtree generated by minimizing the pairwise mutual information [7] among the input variables $\{F_1, F_2, F_3, F_4, C\}$ (left side of Fig. 3(a)), and is initialized on a fully factorized PSDD (right side of Fig. 3(a)).

In this work, we propose to initialize the algorithm on a PSDD that encodes the discriminative relationship between the class variable C and the feature variables $\mathbf{F}$ (see Fig.3(b)). This will initially put more weight on learning the (discriminative) relationship between the class and the features before learning the (generative) relationships between features. Specifically. we force the prime of the root node (in red), to be the class variable C. The sub of this root node is a vtree learned on the feature variable set $\{F_1, F_2, F_3, F_4\}$ (in blue). We then initialize the algorithm on the shown PSDD, where feature variables are independent from each other, but conditioned on the class variable.

Figure 3: Initial model for learning algorithm. a) Vtree learned from dataset and corresponding fully factorized PSDD. b) The root node prime in this vtree is the class variable. In the corresponding PSDD, features are independent from each other but conditioned on the class.

Sensor interface and precision scaling Subsequent scaling stages within the framework of Fig. 2 are determined by the task of interest and by the hardware's capabilities. When the task is classification, the *Scale AC Complexity* stage can be followed by the *Scale Sensor Interfaces* stage, where features and sensors are pruned (see Algorithms 1 and 2), as discussed in detail in [4]. Note that Algorithm 1 includes a step that facilitates model pruning (Algorithm 2), by pre-computing operations corresponding to non-observed variables, whose indicators will always be equal to 1. Since pruned features are removed from the model, this stage might not suitable for density estimation, and can be bypassed. The last scalability stage, *Scale Precision*, consists on scaling down the number of bits used for representation and arithmetic. We consider standard IEEE representations of (64,32,16 and 8 bits). This stage can also be bypassed if the hardware does not support such representations.

Pareto configuration extraction The final stage in Fig. 2 extracts the Pareto optimal configurations from the trade-off space explored in the previous steps (see Algorithm 3). Note that accuracy terms acc can be replaced with other performance metric of interest, such as e.g. log-likelihood for density estimation tasks, or F-score for classification tasks that are sensitive to precision-recall trade-offs.

4 Experimental results

We showcase the proposed framework on a binary classification task on a Human Activity Recognition dataset [1]. In particular, we consider the task of identifying "walking downstairs" from other activities. We binarize the available features and show the results on a 70 binary-feature subset (from correlation-based feature selection) that has undergone a 75% , 10%, 15% train-validation-test random split. Moreover, we consider 20 features for the pruning stage of the framework. We estimate the hardware-aware cost in terms of normalized energy consumption based on post-synthesis energy estimations in 65 nm CMOS technology. Energy consumption scaling in function of number of bits is based on [10]. Figs.4(a),(b) and (c) show the *Scale AC Complexity*, *Scale Sensor Interfaces* and *Scale Precision* steps of the framework, respectively. Lines with circular markers denote the results from initializing the PSDD learner with a fully factorized model (gen. PSDD as in Fig.3 (a)), and lines with cross markers denote the learner initialized with the constrained model (discr.-gen. PSDD as in Fig.3(b). Note that the discr.-gen. PSDD learning approach results in overall higher accuracy, as the discriminative relation is explicitly taken into consideration in the model. However, since the purely generative approach results in cheap but "useful" (accuracy>90%) models, we propose to consider the combination of both Pareto curves: in the final Pareto set, the first three models come

from the generative case and the rest from the discriminative-generative case. Fig. 4(d) shows the resulting accuracy on the test set, as well as robustness tests (in magenta and green), where sets of features of sizes jFj=10,jFj=5,jFj=2, randomly fail. This shows that our framework remains robust, regardless of the learning initialization style. Finally, Figs.4(e) and (f) show the cost vs. test-set log-likelihood trade-off on the "plants" and "accidents" datasets, used commonly for benchmarking density estimation tasks [8, 12]. These results utilize the *Scale AC Complexity* and *Scale Precision* steps of the proposed framework, and demonstrate that density estimation tasks can also benefit from our framework. In both cases, log-likelihood is preserved with savings of more than 50% on C_{HA}.

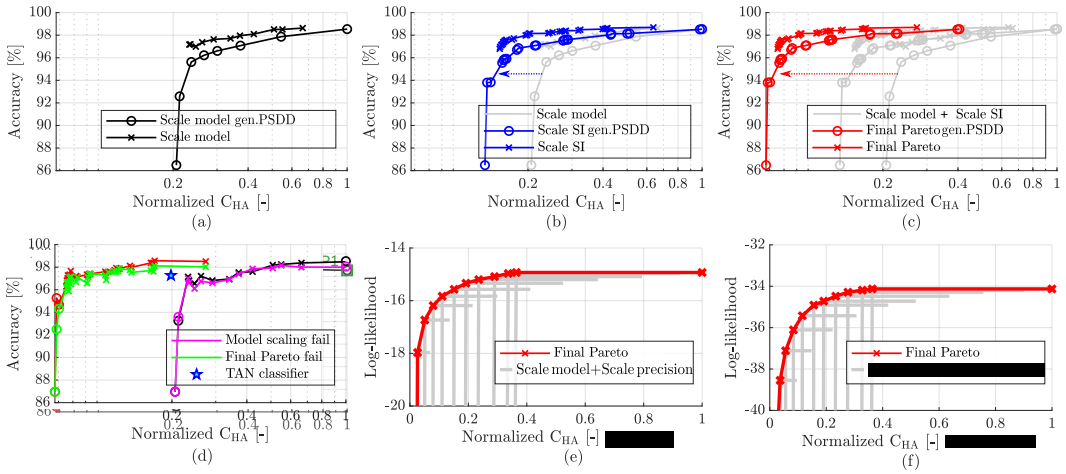

Figure 4: Experimental results: (a,b,c,d) from classification and (e,f) from density estimation tasks.

5 Conclusions and future work

This work discusses recent developments on hardware-awareness for probabilistic frameworks. It also offers insight into newly developed and in-progress techniques that extend their applicability range and improve their performance. In the future, we plan to benchmark the techniques introduced herewith to other datasets and applications, as well as comparing the results with other baselines.

Acknowledgements This work is partially supported by the EU ERC Project Re-SENSE under Grant ERC-2016-STG-71503; NSF grants #IIS-1633857, #CCF-1837129, DARPA XAI grant #N66001-17-2-4032, NEC Research, and gifts from Intel and Facebook Research.

References

[1] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *Proceedings of the 21st ESANN*, 2013.

[2] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

[3] L. Galindez, K. Badami, J. Vlasselaer, W. Meert, and M. Verhelst. Dynamic sensor-frontend tuning for resource efficient embedded classification. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 8(4):858–872, 2018.

[4] L. I. Galindez Olascoaga, W. Meert, N. Shah, M. Verhelst, and G. Van den Broeck. Towards hardware-aware tractable learning of probabilistic models. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, Dec. 2019.

[5] S. Holtzen, T. Millstein, and G. Van den Broeck. Symbolic exact inference for discrete probabilistic programs. In *Proceedings of the ICML Workshop on Tractable Probabilistic Modeling (TPM)*, jun 2019.

[6] D. Kisa, G. V. den Broeck, A. Choi, and A. Darwiche. Probabilistic sentential decision diagrams. In *Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2014.

[7] Y. Liang, J. Bekker, and G. Van den Broeck. Learning the structure of probabilistic sentential decision diagrams. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

[8] D. Lowd and J. Davis. Learning markov network structure with decision trees. In *2010 IEEE International Conference on Data Mining*, pages 334–343. IEEE, 2010.

[9] H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 689–690. IEEE, 2011.

[10] N. Shah, L. I. G. Olascoaga, W. Meert, and M. Verhelst. Problp: A framework for low-precision probabilistic inference. In *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019.

[11] M. Trapp, R. Pehraz, H. Ge, F. Pernkopf, and Z. Ghahramani. Bayesian learning of sum-product networks. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, Dec. 2019.

[12] J. Van Haaren and J. Davis. Markov network structure learning: A randomized feature generation approach. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[13] A. Vergari, N. Di Mauro, and G. Van den Broek. Tutorial slides on tractable probabilistic models. *Conference on Uncertainty in Artificial Intelligence (UAI 2019). Tel Aviv, Israel.*, July 2019.