Extending naive Bayes with precision-tunable feature variables for resource-efficient sensor fusion

Laura Isabel Galindez Olascoaga[1] and **Wannes Meert**[2] and **Herman Bruyninckx**[3] and **Marian Verhelst**[1]

Abstract. Resource-constrained ubiquitous sensing devices suffer from the fundamental conflict between their limited hardware resources and the desire to continuously process all incoming sensory data. The data's representation quality has an immediate impact on both aspects. This paper strives to enable resource-aware and resource-tunable inference systems, which are capable of operating in various trade-off points between inference accuracy and resource usage. We present an extension to naive Bayes that is capable of dynamically tuning feature precision in function of incoming data quality, difficulty of the task and resource availability. We also develop the heuristics that optimize this tunability. We demonstrate how this enables much finer granularity in the resource versus inference accuracy trade-off space, resulting in significant resource efficiency improvements in embedded sensor fusion tasks.

1 INTRODUCTION

The Internet of Things (IoT) paradigm is on the rise, promising an important contribution to tackling societal challenges through improved distributed sensing capabilities. This paradigm is expected to significantly impact application scenarios like e-health and domotics, which already benefit from the widespread availability of embedded sensing devices (i.e., smartphones, activity trackers and service robots) that can reliably gather and process a massive amount of data. The main enabling factor of this vision is the ability to seamlessly integrate several technological solutions which motivates the desire to run complex inference tasks in-situ and in a distributed manner. Yet, smart embedded devices' processing abilities are held back by their limited resource availability, both in terms of energetic as well as computational resources [2]. This creates a fundamental conflict between the desire to fuse information from more and more always-on sensors in embedded devices, and the inability of these embedded devices to process all incoming data continuously and at high precision. This conflict is currently circumvented by running most sensor fusion and sensory inference tasks in the cloud [1, 4]. Yet, this has important consequences towards the system's latency and the user's privacy [5]. Moreover, it does not solve the excessive power consumption spent by the always-on sensors and the wireless link [8].

Efficiently running inference tasks on the devices themselves calls for awareness of the real-time embedded platform's resource limitations. This is in sharp contrast with most state-of-the-art inference approaches, which focus on maximizing information gain and inference accuracy [4] without taking the actual hardware footprint of online inference into account. To facilitate effective sensory fusion inference tasks inside embedded devices, this paper strives to enable resource-aware and resource-tunable inference systems, which are capable of operating in various trade-off points between inference accuracy and resource usage. Such performance-tunability can be realized by dynamically reallocating resources across sensory features in accordance to the task relevance and complexity. Recent techniques, such as feature-cost aware inference [4, 7], perform hardware-cost aware feature selection to minimize the overall resource cost of feature extraction. Additionally to feature-cost one can also adapt known machine learning models such that they are efficiently run on embedded systems by preferring integer operators [25], considering the trade-off between number of operations and accuracy [19], reducing the precision [29] and the value range [9] of the features , or decomposing the model and distributively running the inference task on different processing units [14, 16, 10]. In addition, recent efforts have attempted to integrate such machine learning models and techniques under embedded hardware efficient frameworks [14]. These optimization techniques result in a fixed resource usage versus performance operating trade-off and, as a result, fail to exploit all the resource saving opportunities that the hardware platform can provide. To overcome these limitations, this paper introduces the following innovations:

1. *Feature precision-tunability:* Instead of only selecting or deselecting a sensory feature, embedded platforms can also tune the precision of sensors and sensory feature extraction (i.e. by changing their resolution or number of bits) in return for hardware resource savings through techniques such as approximate computing and approximate sensing. This allows them to dynamically trade-off feature quality for resource efficiency. In this paper, we will extend the naive Bayes fusion model to such feature-precision tunability.
2. *Run-time accuracy-resource-awareness:* Instead of offline feature or precision selection, a dynamic approach should allow run-time accuracy-resource tunability. This requires the creation of a single fusion model, capturing all feature precision-tunability states, which can be explored and traversed at run-time. The resulting model enables run-time adaptations of feature precision in function of incoming data quality, in function of the difficulty of the inference task, or in function of the instantaneous resource availability in the embedded system.

We present an extension to naive Bayes that is capable of dynamic feature precision tunability, as well as heuristics to optimize this tunability. We demonstrate how this enables much finer granularity in

[1] MICAS – Department of Electrical engineering, KU Leuven, email: Laura.Galindez@esat.kuleuven.be
[2] DTAI – Department of Computer Science, KU Leuven
[3] PMA – Department of Mechanical Engineering, KU Leuven

[4] In this paper we refer to inference accuracy as the percentage of correctly predicted queries from a test set.

the resource versus inference accuracy trade-off space, resulting in significant resource efficiency improvements in embedded sensor fusion tasks. These performance tuning capabilities are of up most relevance in data-dense and resource-constricted environments like the IoT. Therefore, we demonstrate the functionality of our proposed techniques in sensor fusion datasets related to applications relevant to this paradigm.

The paper is structured as follows. In Section 2 we give a theoretical background of naive Bayes classifiers and Bayesian Networks (BN) and we explain how precision tuning enables resource-efficiency in the current context. Section 3 gives the details of the proposed feature precision-tunable BN and explains how parameter learning and inference are performed in it. In Section 4 we propose a heuristic that uses the proposed BN to select optimal operating points in the resource versus inference accuracy space and we evaluate the trade-off it achieves by performing experiments on four data corpora in Section 5. Finally, we discuss the results as well as our contributions and future work in Section 6.

2 BACKGROUND

2.1 Bayesian Networks

Bayesian Networks (BN) are directed acyclic graphs that compactly encode a joint probability distribution over a set of random variables [24]. Consider a set of random variables $\mathbf{U} = \{F_1, \ldots, F_n\}$ where each feature F_i may take values from a finite set $Val(F_i)$. A Bayesian Network for a set of random variables $\mathbf{U}$ is formally defined as the pair $B =< G, \Theta >$. The first component G represents the graph which encodes conditional independence assumptions. The nodes represent variables F_i and its arcs represent the probabilistic dependencies between variables. The second component Θ represents the set of conditional probability distributions $\Pr(F_i|\Pi_{F_i})$ that quantify the network where Π_{F_i} denotes the parents of F_i.

The joint probability distribution defined by the network B is given by

$$Pr(F_1, ..., F_n) = \prod_{i=1}^{n} Pr(F_i|\Pi_{F_i}). \quad (1)$$

Inference in BNs, $\Pr(\mathbf{Q}|\mathbf{E} = \mathbf{e})$, can be performed by assigning values $\mathbf{e}$ to variables $\mathbf{E}$ that are observed and by summing out variables $\mathbf{U}\backslash(\mathbf{Q} \cup \mathbf{E})$ that are not part of the query $\mathbf{Q}$.

2.2 Bayesian Network Classifiers

Classification is the task of assigning a class label C to instances described by a set of features $F_1, ..., F_n$. Such a task can be tackled by a Bayesian network where one of the random variables, C, is considered the class and the other random variables, $F_1, \ldots, F_n$, represent the features. The task is now to find the most likely value for the class variable C:

$$c = \arg\max_c \Pr(C = c|F_1 = f_1, \ldots, F_n = f_n), \quad (2)$$

where c is the current class and f_i is the observed value for feature F_i.

A widely used type of Bayesian classifier is the naive Bayes classifier [12]. The main assumption is that every feature is independent of the other features given that the class is known. The graphical structure of the naive Bayes network is shown in Figure 1. This assumption allows for efficient learning and inference as it simplifies Equation 2 to

$$c = \max_c \Pr(F_1 = f_1|C = c) \ldots \Pr(F_n = f_n|C = c) \Pr(C = c)$$

by applying the rule of Bayes and conditional independence. Despite the independence assumption, naive Bayes classifiers perform surprisingly good. This makes them one of the most effective and efficient inductive learning algorithms [33, 6].

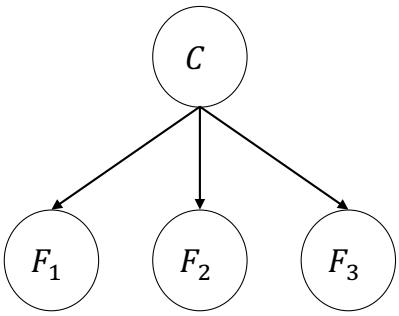

Figure 1. Graphical representation of the naive Bayes classifier

2.3 Resource awareness and precision tuning

Embedded hardware platforms have to operate under very scarce resources. First and foremost, their miniaturization results in very limited battery capabilities which motivates the quest for high energy efficient designs and methodologies. Moreover, due to size, cooling and cost restrictions, the computational bandwidth of these devices is extremely scarce. This has sparked an enormous amount of research into adaptive hardware over the last decade. Under this paradigm, resource consumption can be tuned at run-time to be lower at the expense of reduced quality sensor streams or computations. This dynamic trade-off is achievable in several ways:

1. *Noisy sensors:* The amount of noise present in sensory measurement strongly depends on the amount of energy spent in the sensor front-end (in its filters, amplifiers, etc). By tolerating more statistical noise on the measurement result, energy can be saved [15, 3].
2. *Stochastic computing*: The resulting accuracy of digital computations can be traded off against processing resource usage and energy consumption by using stochastic or approximate computing techniques. In stochastic computing, for example, numbers are represented by bit-streams that can be processed by very simple circuits such as standard logic gate arrays. These implementations allow a limited amount of errors (stochastic noise) in the digital embedded calculations in return for a more efficient implementation [21].
3. *Reduced precision sensing and computing:* Instead of applying aforementioned stochastic techniques to dynamically trade feature quality for resource savings, significant resource savings are also achievable by simply limiting the amount of bits with which sensor values are sampled and digitally processed for feature extraction. Standard hardware platforms digitize sensory values at fixed precision and typically process them with 16-bit resolution. Recent works present the development precision-tunable digitizers, as well as digital processing platforms capable of dynamically

adjusting the precision with which internal computations are performed [23, 20].

Within this paper, we focus on enabling the feature quality versus resource trade-off through the latter technique: computations with variable precision features. The results are transferable to the discussed alternatives, which is left for future work. Under variable precision computations, the extracted features $\mathbf{U} = \{F_1, ..., F_n\}$ are each computed and represented by a tunable amount of bits. The number of bits representing the feature, directly impacts the set of values a feature F_i can take on and will be referred to as $F_{i,m}$ with m the number of bits. More specifically, when using a m-bit representation, the feature can take $|\,Val(F_{i,m})| = 2^m$ possible values.

As we are interested in studying resource efficiency in sensory applications we construct these m-bit representations by following a signal processing quantization approach [27]. Mapping of the original feature to an m-bit representation is based on comparisons with decision levels t_k. If the feature value is between t_k and t_{k+1} it gets mapped to a quantization level l_k, where the number of levels must be equal to 2^m. In this paper, the decision levels t_k are derived from the feature value range and the set of lower precision decision levels is a subset of the higher precision decision levels (see also Section 3).

State-of-the-art implementations show that under such computational precision tuning, the resource cost (here expressed in terms of energy per computational operation) scales more than quadratically with the computational precision, expressed in terms of number of bits [22]. In this paper, we will assume that the feature cost (denoted $T_{i,m}$) of feature F_i computed with precision m-bits is equal to

$$T_{i,m} = \alpha_i \cdot m^2. \tag{3}$$

where α_i is the feature-dependent cost of the nominal precision feature.

3 VARIABLE FEATURE PRECISION NAIVE BAYES

Tuning feature precision enables a wide range of resource dependent operation points.We make the probabilistic relations required by classification tasks explicit in a Bayesian Network (BN) where features can be observed at different precision levels.

3.1 Model Structure

The proposed BN represents a naive Bayes classifier that has multiple versions of the same feature in each leaf, as shown by Figure 2.

Feature versions with the highest precision ($F_{i,m}$) are directly linked to the class variable C. There are no direct links between lower precision feature versions $F_i \setminus F_{i,m}$ and the class variable since the relation between these versions is deterministic. The proposed structure encodes the following joint probability distribution over the multiple feature version sets $F_i = \{F_{i,m}, \ldots, F_{i,0}\}$ and the class variable C

$$Pr(C, F_{1,m}, ..., F_{n,0}) = \prod_{i=1}^{n} \prod_{b=0}^{m-1} Pr(F_{i,b}|F_{i,b+1}) \cdot Pr(F_{i,m}|C) \cdot Pr(C). \tag{4}$$

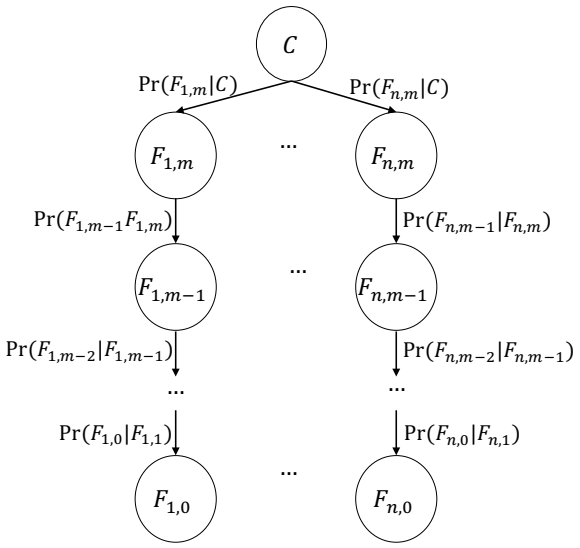

Figure 2. Naive Bayes model extended with multiple feature quality versions

3.2 Parameter Learning

We assume that the studied features were generated by a continuous distribution [24], therefore, we model the conditional probabilities between features of highest precision and classes $Pr(F_{i,m}|C)$ as Gaussian distributions [18]. Even though the model includes $n \times (m + 1)$ parameters θ, only the conditional probabilities between features of highest precision and classes $Pr(F_{i,m}|C)$, must be trained since the conditional probabilities between lower precision features $Pr(F_{i,b}|F_{i,b+1})$ are deterministic. Once we have knowledge of the decision levels t_k that generated the lower precision features, we are able to systematically add them to the tails of the proposed naive Bayes structure.

3.3 Inference

At any given time, every feature F_i is observed at only one of the precision options b_i depending on the current resource consumption desires and constraints. Given observation $o = \{f_{1,b_1}, f_{2,b_2}, ..., f_{n,b_n}\}$, classification is performed by estimating the class posterior probability given by

$$Pr(C|o) \sim \prod_{i=1}^{n} Pr(fi, b_i|C) \cdot Pr(C). \tag{5}$$

This implies that, for every observed feature, its lower precision versions are not observed, while their higher precision versions are marginalized.

Consider the example depicted in Figure 3 which may correspond to a robot navigation application, as discussed in Section 5.2. Suppose we obtain sensor readings at 8 bit, 4 bit and 2 bit for sensors $S1$, $S2$ and $S3$, respectively and we decide to turn off sensor $S4$. Here, we can estimate the class posterior probability (which can be a location, for example) with the following equation

$$Pr(C|S_{1,8b}, S_{2,4b}, S_{3,2b}) \sim Pr(S_{1,8b}|C) \cdot \sum_{S_{2,8b}} Pr(S_{2,4b}|S_{2,8b}) Pr(S_{2,8b}|C) \cdot \sum_{S_{3,4b}} \sum_{S_{3,8b}} Pr(S_{3,2b}|S_{3,4b}) \cdot Pr(S_{3,4b}|S_{3,8b}) \cdot Pr(S_{3,8b}|C) \cdot Pr(C), \quad (6)$$

and predict the class $c \in C$ with the highest posterior.

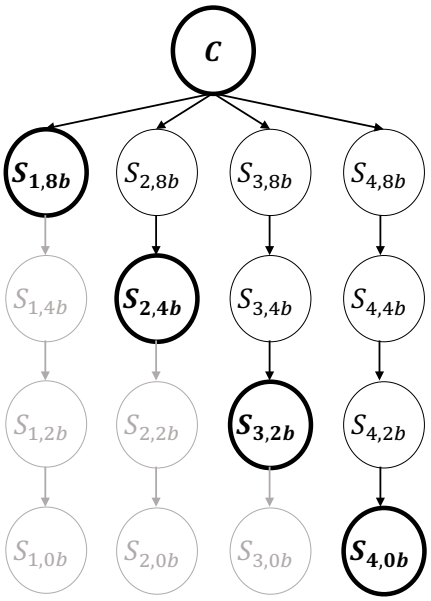

Figure 3. Example of a four feature application where each of them is observed at a different precision (circles with bold edges). Features with higher precision than the observed are marginalized (circles with black edges) and versions with lower precision are not observed (circles with gray edges). Note that feature 4 is observed with "0 bit" precision, which is equivalent to pruning it.

4 ACCURACY-RESOURCE TUNABILITY

The proposed model enables multiple resource and accuracy dependent operating points. In this paper we analyze the trade-off induced by the available feature combination choices and we propose a methodology to find the optimal operating points given the system's resource constraints.

We propose an accuracy-resource sensitive algorithm that selects the optimal feature precision across the accuracy-resource usage trade-off space. At each iteration, we select the feature set that optimizes a cost function CF, which is defined according to the desired application and the constraints thereto [13, 17]. In this paper we maximize the cost function given by

$$CF = \log\left(\frac{\Delta resource}{\max(resource)}\right) - \log(\Delta accuracy), \quad (7)$$

where the term Δ refers to the predicted state difference between time k and time k+1 as will be detailed in Algorithm 1. The greedy neighborhood search in our heuristic ensures resource reduction, and the cost function further motivates it by explicitly trading off the two terms.

Two of this algorithm's aspects distinguish it from conventional state-of-the-art feature selection techniques [7, 28, 31]: 1) We merge accuracy gain and resource usage in a joint cost optimization, hence taking hardware implementation aspects into account from the algorithmic level. 2) In contrast to cost-aware feature selection techniques which decide whether to use a feature or not, we enable the selection of a variety of feature precision combinations.

Algorithm 1 details the method. We initialize the selected feature set to the highest precision feature combination $\mathbf{U}_{selected} = \{F_{1,m}, ..., F_{n,m}\}$. At each iteration, we perform a greedy neighborhood search over n feature combination candidates. In each candidate i, the precision of feature F_i is dropped one level with respect to the current precision. We evaluate the classification accuracy and resource usage of each candidate and select the one that maximizes the cost function CF. The procedure is repeated until the feature combination with the lowest precision is selected ($\mathbf{U}_{selected} = \{F_{1,0}, ..., F_{n,0}\}$). Note that the algorithm is able to perform feature pruning if a "null precision" leaf is added to the Naive Bayes model (see Figure 3 for an example).

Classification accuracy is computed by estimating the posterior probability $Pr(C|k)$ of every instance k from a testing data-set $\mathbf{U}_{test}$ and comparing the prediction to to the instance's label (see Algorithm 2).

5 EXPERIMENTS

We evaluate the resource-accuracy trade-off achieved by our proposal with one synthetic dataset and three data corpora from two real applications relevant to the IoT paradigm.

5.1 Synthetic Data

This dataset consists of 2000 points sampled from 4 Gaussians $\mathcal{N}(\mathbf{m}_i, \boldsymbol{\sigma}_i)$, $i = \{1, 2, 3, 4\}$, where $\mathbf{m}_1 = \begin{pmatrix} -1.66 \\ -0.33 \\ -0.33 \\ -2.00 \end{pmatrix}$, $\mathbf{m}_2 = \begin{pmatrix} 1.00 \\ 0.5 \\ 1.00 \\ 1.00 \end{pmatrix}$, $\mathbf{m}_3 = \begin{pmatrix} 3.33 \\ 2.00 \\ 0.5 \\ 0.5 \end{pmatrix}$, $\mathbf{m}_4 = \begin{pmatrix} -1.66 \\ -1.43 \\ -0.66 \\ -3.33 \end{pmatrix}$, $\boldsymbol{\sigma}_1 = \begin{pmatrix} 0.80 \\ 1.00 \\ 1.00 \\ 1.00 \end{pmatrix}$, $\boldsymbol{\sigma}_2 = \begin{pmatrix} 0.70 \\ 1.00 \\ 1.00 \\ 1.00 \end{pmatrix}$, $\boldsymbol{\sigma}_3 = \begin{pmatrix} 1.00 \\ 1.00 \\ 1.00 \\ 1.00 \end{pmatrix}$ and $\boldsymbol{\sigma}_4 = \begin{pmatrix} 1.00 \\ 1.00 \\ 1.00 \\ 1.00 \end{pmatrix}$. The Gaussians are defined to have different degrees of overlap in every dimension and have therefore a varying miss-classification risk for different feature combinations. We quantize the data-set at 5, 3, 2 and 1 bits and randomly divide it into a training and a testing set (used for model training and accuracy estimation, respectively). We compute the resource usage with Equation 3, as included in Table 1 . To assign the variable α_i, we assume that features that are less likely to cause miss-classification would be more expensive to extract and compute in a real application. Thus giving a higher value to them. We add a "null precision" leaf, to enable feature pruning as shown in the example depicted by Figure 3.

Figure 4 shows the resource vs accuracy trade-off curve achieved by the proposed algorithm and achieved by a typical resource-aware heuristic [5] in red and blue, respectively. The gray point cloud represents all the possible accuracy-resource trade-off operational points to select from. The proposed heuristic has a richer feature combination space to select from, which prevents accuracy degradation for a

[5] The typical resource-aware heuristic considers only features at the highest precision $F_{i,m}$ and decides whether to prune them by maximizing CF.

Algorithm 1: Feature precision selection algorithm for accuracy-resource tradeoff

Feature precision selection ($\mathbf{U}_{test}$, $T_{i,m}$, Θ, C, CF);
Input : $\mathbf{U}_{test}$, $T_{i,m}$, Θ, C, CF
Output: Selected feature set $\mathbf{U}_{selected_k} = \{F_{1,b_1}, ..., F_{n,b_n}\}$
k=0;
/* Initialize with the highest precision feature set */
$\mathbf{U}_{selected_k} = \{F_{1,m}, ..., F_{n,m}\}$
$accuracy_k \leftarrow$ AccuracyEvaluation(Θ,C, $\mathbf{U}_{selected_k}$)
$resource_k \leftarrow T_{i,b_i} \; \forall \; F_{i,b_i} \in \mathbf{U}_{selected_k}$
/* while the lowest feature precision has not been selected */
while $\boldsymbol{U}_{selected_k} \neq \{F_{1,0}, ..., F_{n,0}\}$
do
  for $i = 1$ *to* n // For each candidate combination
  do
    /* drop F_i's precision one level */
    $\mathbf{U}_{candidate_i} \leftarrow \mathbf{U}_{selected_k} \setminus F_{i,b_i} \vee \{F_{i,b_i-1}\}$
    $accuracy_{candidate_i} \leftarrow$ AccuracyEvaluation(Θ, C, $\mathbf{U}_{candidate_i}$);
    $resource_{candidate_i} \leftarrow T_{i,b_{canditate_i}} \; \forall \; F_{i,b_{candidate_i}} \in \mathbf{U}_{candidate_i}$;
    $\Delta accuracy_{candidate_i} = accuracy_k - accuracy_{candidate_i}$;
    $\Delta resource_{candidate_i} = resource_k - resource_{candidate_i}$;
  end
  update k=k+1;
  $\mathbf{U}_{selected_k} \leftarrow \underset{\mathbf{U} \in \mathbf{U}_{candidate}}{\text{argmin}} \; \text{CF}(\Delta accuracy_{candidate}, \Delta resource_{candidate})$;
  update $accuracy_k \leftarrow$ AccuracyEvaluation(Θ,C, $\mathbf{U}_{selected_k}$)
  update $resource_k \leftarrow T_{i,b_i} \; \forall \; F_{i,b_i} \in \mathbf{U}_{selected_k}$
  Return: $\mathbf{U}_{selected_k}$
end

Algorithm 2: Classification accuracy evaluation algorithm

AccuracyEvaluation (Θ,C,$\mathbf{U}$);
Input : Θ,C,$\mathbf{U}$
Output: accuracy
$correct = 0$;
for $k = 1$ *to* N // With N the number of instances in the testing set
do
  /* For each class we approximate the posterior probability given the instance currently analyzed */
  $Pr(C|k) \leftarrow Pr(k|C) \cdot Pr(C)$
  /* We predict the class with the highest posterior probability */
  $c_{max_k} = \underset{c \in C}{\text{argmax}} Pr(C|k)$
  if $c_{max_k} == c_k$ **then** correct=correct+1;
end
update $accuracy \leftarrow correct \div N$;
Return: accuracy

resource usage scale-down of up to 20 times. The non-tunable precision heuristic has comparatively very few feature combination options to select from, which leads, in contrast, to a maximum resource scaling of approximately 2 times without accuracy degradation.

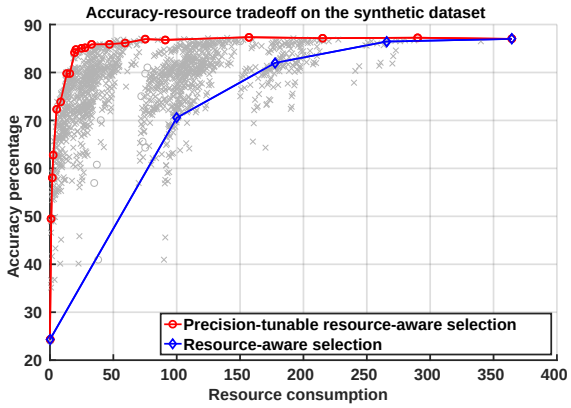

Figure 4. Algorithm performance comparison on the synthetic dataset.

5.2 Real Datasests

We analyze three sensor based applications that benefit from feature precision tuning. The first is a robot navigation task. The second and third dataset are activity recognition tasks.

Wall-Following Robot Navigation We analyze a public domain dataset that was collected as a mobile robot navigates through a room following the wall in a clockwise direction, for 4 rounds, using 4 ultrasound sensors positioned on the front, left, right and back of its body [11]. Four states can be identified from the sensor readings: Move-Forward, Slight-Right-Turn, Sharp-Right-Turn or Slight-Left-Turn. The data-set has a precision of 8 bits and we further quantize it at 5, 2 and 1 bits. We assume the four sensors have the same hardware properties, so we set the variable α_i equal to one for all of them and use Equation 3 to generate the resources for the experiments, as shown in Table 1. Furthermore, we add a random number between 0 and 5 to each cost to simulate non ideal performance conditions.

Figure 5 shows the cost-accuracy trade-off achieved by the proposed precision-tunable heuristic and the trade-off achieved by a cost aware method in red and blue, respectively. The gray crosses represent all the possible operation points to choose from. Both heuristics display negligible accuracy degradation when their resource consumption is scaled down a factor 2 (from 400 to 200). The slight accuracy gain achieved by the non-tunable heuristic can be due to the discretization related improvements discussed in [32] and [29] . For a factor 4 resource scaling (from 400 to 100) the non-tunable heuristic has already pruned 3 out of its four features which causes the accuracy to degrade from 90% to 75%. The precision-tunable heuristic keeps observing all features, yet reduces their precision which also produces a factor 4 resource consumption downscale but no accuracy degradation.

USC-HAD This dataset was designed as a benchmark for Human Activity Detection (HAD) algorithm comparisons [34]. It was collected by an Inertial Measurement Unit (IMU) placed in the subjects'

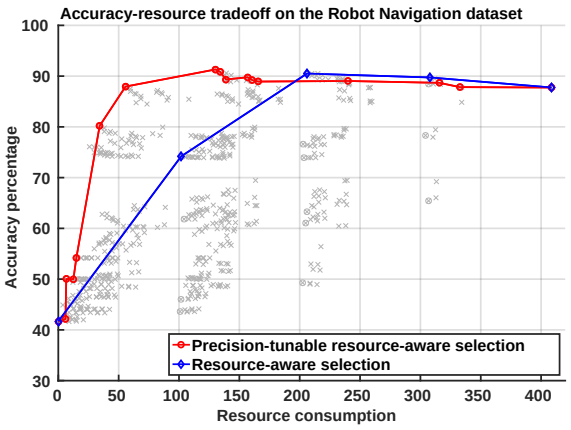

Figure 5. Performance comparison in the robot navigation application

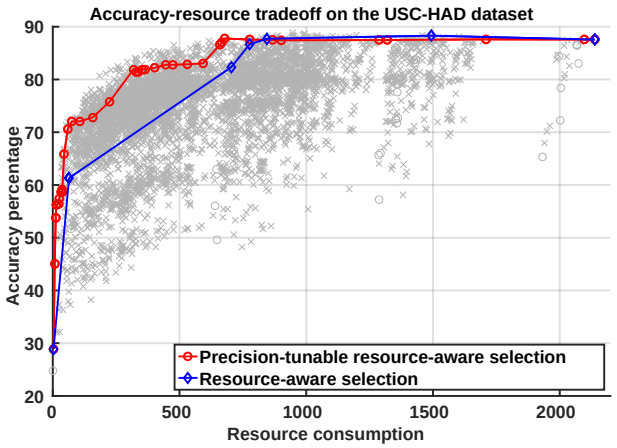

Figure 6. Trade-off comparison on the Human Activity Detection dataset with gyroscope and accelerometer.

hip consisting of a 3-axis accelerometer and a 3-axis gyroscope and it contains measurements for the identification of 12 different low-level daily activities. In accordance to previously performed Activity Recognition analyses [7, 26], the activities that can be best classified with naive Bayes and that are therefore used in this experiment are Walking-forward, Running-Forward, Sitting and Sleeping.

We tuned the dataset's precision from the original 8 bits to 5,4,3,2 and 1 bits. For resource assignment, we consider that the power consumption of a gyroscope can be up to 10 times that of an accelerometer [34] so we set the corresponding α_i variables to 1 and 10, respectively, and use Equation 3 to calculate the resource consumption. Like in the previous experiment, we add a random number between 0 and 5 to simulate non ideal behavior. The resource consumption assignments for this experiment are detailed in Table 1. Again, we enable feature pruning through the addition of the 0 bit leaf to he model.

Figure 6 shows the cost-accuracy trade-off curves achieved by the precision-tunable and the cost-aware only heuristics in red and blue, respectively. The possible operating points are represented by gray crosses. For a resource consumption downscale of 2.5 (from 2130 to 845), the non-precision tunable heuristic suffers from an accuracy degradation of 6% (88% to 82%), while there is no accuracy reduction with the precision-tunable method. Although the 6% accuracy loss/2x cost saving of the non-tunable strategy could be acceptable in some situations, it is worth noting the limitations imposed by the available number of operating points. In addition to the 2.5x resource downscale, only a 30x reduction (from 2130 to 65) is possible at the expense of accuracy degrading from 88% to 61%. The precision-tunable strategy has, in contrast, the possibility to choose from approximately 26 operation points, with up to 6x resource savings before accuracy is lower than 80%.

HAR-RIO In this dataset, 5 activities (Sitting-Down, Standing-Up, Standing, Walking, and Sitting) can be identified from 8 hours of recordings performed by 4 accelerometers positioned in the waist, left thigh, right ankle and right arm of 4 healthy subjects [30]. The accelerometers are tri-axial which results in a total number of 12 features; $\{x_i, y_i, z_i\}$, $i = \{1, 2, 3, 4\}$. For the experiments in this paper, 9 of those features were selected in accordance to previously performed classification algorithm benchmarking [30], namely $\{y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_4, y_4, z_4\}$. The dataset's precision is 8 bits, we quantize it at 4,3,2,1 bits and we add the "null precision" leaf that enables feature pruning.The resource parameters used in this experiment are listed in 1.

Figure 7 shows the results from this experiment with the same color coding as previous. The precision-tunable approach's performance is superior, as it achieves up to 12x resource savings (from 620 to 50) for a maximum accuracy degradation of 4% (from 80% to 76%). The non-tunable strategy displays accuracy losses of less than 5% up to a resource consumption scaling of 3x (620 to 200). For any resource down scaling larger than that, the accuracy degrades more than 10%.

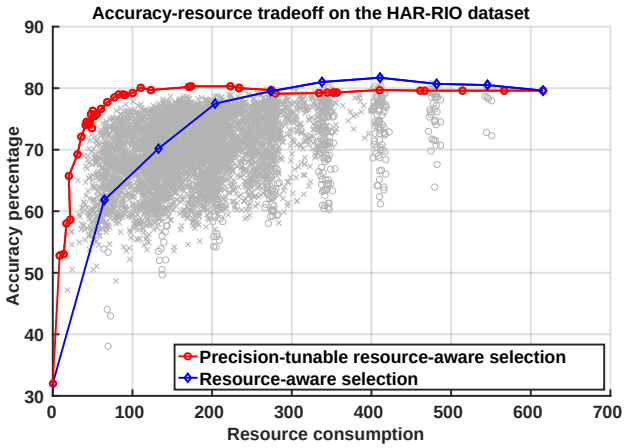

Figure 7. Trade-off comparison on the Human Activity Recognition dataset with accelerometers.

6 CONCLUSIONS AND DISCUSSION

Our main contribution in this paper was to enable efficient embedded sensor fusion through a resource-aware naive Bayes model, capable of exploiting variable precision features. By encapsulating various precision features within the model structure, we enable the possibility to dynamically tune resource consumption and inference accuracy according to the circumstances and available resources. We propose an algorithm that finds optimal operating points by reducing resource consumption and minimizing accuracy degradation.

Table 1. Feature resources used for experiments

		m bits feature precision						
Dataset	α	10	8	5	4	3	2	1
Synthetic								
Feat. 1	1	100	-	25	-	9	4	1
Feat. 2	0.7	70	-	17.5	-	6.3	2.8	0.7
Feat. 3	0.9	90	-	22.5	-	8.1	3.6	0.9
Feat. 4	0.8	80	-	20	-	7.2	3.2	0.8
Robot	1	100	-	25	-	-	4	1
USC-HAD								
Accel.	1	-	64	25	16	9	4	1
Gyro.	10	-	640	250	160	90	40	10
HAR-RIO	1	-	64	25	16	9	4	1

We have compared our scheme with a state-of-the-art resource-aware feature selection technique and we conclude that overall our scheme has better cost saving capabilities due to the rich variety of operational points it can choose from. We tested one artificial and three public domain data corpora with the proposed methodology. Accuracy degradation was prevented while achieving resource usage scalings of 20x for the synthetic dataset, 4x for the Robot Navigation application, 6x for the Human Activity Detection application with accelerometers and gyroscopes, and 12x for the Human Activity Recognition application with accelerometers. The non-tunable precision heuristic achieved, in comparison, a resource scaling of 2x for the synthetic dataset, 2x for the Robot Navigation application, 2.5x for the Human Activity Detection application with accelerometers and gyroscopes, and 3x for the Human Activity Recognition application with accelerometers.

There are many ways in which feature quality tuning can improve hardware resource efficiency. We proved this concept by tuning feature precision but the next step in our work will be to extend the proposed method to other quality tuning paradigms beyond precision tunability such as varying levels of noisy sensing. This will potentially require the modification of the proposed multiple-level Bayesian Network as the relationship between nodes of different qualities will not be deterministic anymore. Furthermore, we will explore more complex structures for applications that are not modeled with sufficient accuracy under the naive Bayes independence assumption.

The long term goal is to integrate the optimal feature precision selection scheme in an embedded sensory application, where the system dynamically and autonomously selects features and their precision given the current state of the hardware devices with limited computational overhead. This scheme could enable the seamless integration of sensory based algorithms into smart environments which is one of the elements envisioned for the IoT.

REFERENCES

[1] Rajesh Arumugam, Vikas Reddy Enti, Liu Bingbing, Wu Xiaojun, Krishnamoorthy Baskaran, Foong Foo Kong, A Senthil Kumar, Kang Dee Meng, and Goh Wai Kit, 'Davinci: A cloud computing framework for service robots', in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 3084–3089. IEEE, (2010).

[2] Luigi Atzori, Antonio Iera, and Giacomo Morabito, 'The internet of things: A survey', *Computer networks*, **54**(15), 2787–2805, (2010).

[3] Komail MH Badami, Steven Lauwereins, Wannes Meert, and Marian Verhelst, 'A 90 nm CMOS, power-proportional acoustic sensing frontend for voice activity detection', *Solid-State Circuits, IEEE Journal of*, **51**(1), 291–302, (2016).

[4] F. H. Bijarbooneh, W. Du, E. C. H. Ngai, X. Fu, and J. Liu, 'Cloud-assisted data fusion and sensor selection for internet of things', *IEEE Internet of Things Journal*, **3**(3), 257–268, (June 2016).

[5] Azzedine Boukerche, Antonio AF Loureiro, Eduardo F Nakamura, Horacio ABF Oliveira, Heitor S Ramos, and Leandro A Villas, 'Cloud-assisted computing for event-driven mobile services', *Mobile Networks and Applications*, **19**(2), 161–170, (2014).

[6] Jie Cheng and Russell Greiner, 'Comparing Bayesian network classifiers', in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pp. 101–108. Morgan Kaufmann Publishers Inc., (1999).

[7] Jian Cui and Bin Xu, 'Cost-effective activity recognition on mobile devices', in *Proceedings of the 8th International Conference on Body Area Networks*, BodyNets '13, pp. 90–96, ICST, Brussels, Belgium, Belgium, (2013). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[8] Artem Dementyev, Steve Hodges, Stephen Taylor, and Johan Smith, 'Power consumption analysis of bluetooth low energy, zigbee and ant sensor nodes in a cyclic sleep scenario', in *Wireless Symposium (IWS), 2013 IEEE International*, pp. 1–4. IEEE, (2013).

[9] K. Frank, P. Robertson, S. Fortes Rodriguez, and R. Barco Moreno, 'Faster bayesian context inference by using dynamic value ranges', in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pp. 50–55, (2010).

[10] K. Frank, M. Rckl, and P. Robertson, 'The bayeslet concept for modular context inference', in *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008. UBICOMM '08. The Second International Conference on*, pp. 96–101, (Sept 2008).

[11] A. L. Freire, G. A. Barreto, M. Veloso, and A. T. Varela, 'Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study', in *Robotics Symposium (LARS), 2009 6th Latin American*, pp. 1–6, (Oct 2009).

[12] Nir Friedman, Dan Geiger, and Moises Goldszmidt, 'Bayesian network classifiers', *Journal of Machine Learning*, **29**(2), 131–163, (1997).

[13] Isabelle Guyon and André Elisseeff, 'An introduction to variable and feature selection', *J. Mach. Learn. Res.*, **3**, 1157–1182, (March 2003).

[14] Josué Iglesias, Ana M. Bernardos, Paula Tarrío, José R. Casar, and Henar Martín, 'Design and validation of a light inference system to support embedded context reasoning', *Personal and Ubiquitous Computing*, **16**(7), 781–797, (2012).

[15] Peter Kinget and Michiel Steyaert, 'Impact of transistor mismatch on the speed-accuracy-power trade-off of analog CMOS circuits.', in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 333–336, (1996).

[16] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, L. Jiao, L. Qendro, and F. Kawsar, 'Deepx: A software accelerator for low-power deep learning inference on mobile devices', in *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 1–12, (April 2016).

[17] Steven Lauwereins, Wannes Meert, Jort Gemmeke, and Marian Verhelst, 'Ultra-low-power voice-activity-detector through context- and resource-cost-aware feature selection in decision trees', in *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, (Sept 2014).

[18] Jonathan Lester, Tanzeem Choudhury, Nicky Kern, Gaetano Borriello, and Blake Hannaford, 'A hybrid discriminative/generative approach for modeling human activities.', in *IJCAI*, volume 5, pp. 766–772, (2005).

[19] Daniel Lowd and Pedro Domingos, 'Learning arithmetic circuits', *arXiv preprint arXiv:1206.3271*, (2012).

[20] Bert Moons, Bert De Brabandere, Luc Van Gool, and Marian Verhelst, 'Energy-efficient convnets through approximate computing', in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1–8. IEEE, (2016).

[21] Bert Moons and Marian Verhelst, 'Energy and accuracy in multi-stage stochastic computing', in *New Circuits and Systems Conference (NEWCAS), 2014 IEEE 12th International*, pp. 197–200. IEEE, (2014).

[22] Bert Moons and Marian Verhelst, 'Dvas: Dynamic voltage accuracy scaling for increased energy-efficiency in approximate computing', in *Low Power Electronics and Design (ISLPED), 2015 IEEE/ACM International Symposium on*, pp. 237–242. IEEE, (2015).

[23] Bert Moons and Marian Verhelst, 'A 40nm CMOS, 35mW to 270mW,

precision-scalable cnn processor for run-time scalable embedded vision', in *IEEE VLSI Symposium*. IEEE, (2016).
[24] Judea Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1988.
[25] Nico Piatkowski, Sangkyun Lee, and Katharina Morik, 'Integer undirected graphical models for resource-constrained systems', *Neurocomputing*, **173**, 9–23, (2016).
[26] Olga Politi, Iosif Mporas, and Vasileios Megalooikonomou, 'Human motion detection in daily activity tasks using wearable sensors', in *Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European*, pp. 2315–2319. IEEE, (2014).
[27] John G. Proakis and Dimitris K Manolakis, *Digital Signal Processing (4th Edition)*, Pearson, 2006.
[28] Yvan Saeys, Iaki Inza, and Pedro Larraaga, 'A review of feature selection techniques in bioinformatics', *Bioinformatics*, **23**(19), 2507–2517, (2007).
[29] Sebastian Tschiatschek and Franz Pernkopf, 'On Bayesian network classifiers with reduced precision parameters', *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **37**(4), 774–785, (2015).
[30] Wallace Ugulino, Débora Cardador, Katia Vega, Eduardo Velloso, Ruy Milidiú, and Hugo Fuks, 'Wearable computing: Accelerometers data classification of body postures and movements', in *Advances in Artificial Intelligence-SBIA 2012*, 52–61, Springer, (2012).
[31] Zhixiang Xu, Matt J Kusner, Kilian Q Weinberger, Minmin Chen, and Olivier Chapelle, 'Classifier cascades and trees for minimizing feature evaluation cost', *The Journal of Machine Learning Research*, **15**(1), 2113–2144, (2014).
[32] Ying Yang and Geoffrey I Webb, 'Proportional k-interval discretization for naive-Bayes classifiers', in *ECML*, pp. 564–575. Springer, (2001).
[33] Harry Zhang, 'The optimality of naive Bayes', in *Proceedings of FLAIRS*, (2004).
[34] Mi Zhang and Alexander A. Sawchuk, 'USC-HAD: A daily activity dataset for ubiquitous activity recognition using wearable sensors', in *ACM International Conference on Ubiquitous Computing (Ubicomp) Workshop on Situation, Activity and Goal Awareness (SAGAware)*, Pittsburgh, Pennsylvania, USA, (September 2012).