Towards Resource-Efficient Classifiers for Always-On Monitoring

Jonas Vlasselaer[1], Wannes Meert[2], and Marian Verhelst[1]

[1] MICAS, Dept. of Electrical Engineering, KU Leuven
[2] DTAI, Dept. of Computer Science, KU Leuven

Abstract. Emerging applications such as natural user interfaces or smart homes create a rising interest in electronic devices that have always-on sensing and monitoring capabilities. As these devices typically have limited computational resources and require battery powered operation, the challenge lies in the development of processing and classification methods that can operate under extremely scarce resource conditions. To address this challenge, we propose a two-layered computational model which enables an enhanced trade-off between computational cost and classification accuracy: The bottom layer consists of a selection of state-of-the-art classifiers, each having a different computational cost to generate the required features and to evaluate the classifier itself. For the top layer, we propose to use a Dynamic Bayesian network which allows to not only reason about the output of the various bottom-layer classifiers, but also to take into account additional information from the past to determine the present state. Furthermore, we introduce the use of the Same-Decision Probability to reason about the added value of the bottom-layer classifiers and selectively activate their computations to dynamically exploit the computational cost versus classification accuracy trade-off space. We validate our methods on the real-world SINS database, where domestic activities are recorded with an accoustic sensor network, as well as the Human Activity Recognition (HAR) benchmark dataset.

1 Introduction

There is a rising interest in electronic devices that have always-on sensing and monitoring capabilities [11, 9]. Consider, for example, the task of Ambient Assisted Living (AAL), where the goal is to allow recovering patients and elderly persons to stay in their home environment instead of traveling to a hospital or living in a care center. Crucial to achieve this goal, is a continuous, always-on monitoring setup. Such a setup entails three components: First, wireless sensors and wearables equipped with various sensors are connected in a wireless network to sense and record the physical conditions of the environment. Second, the measurements are processed using signal processing techniques to extract useful features and compress the data without loss of relevant information. Third, machine learning algorithms are used to train a model for the task at hand (e.g. an activity detector). Typically, these three components are analyzed and optimized separately [13, 16, 20]. For the machine learning model, the input features are assumed given and a classifier is trained to maximize its accuracy. The use of battery-powered, computationally-limited embedded devices, however, poses new challenges

as it requires methods that can operate under scarce resource conditions. To address these challenges, we propose to optimize these components simultaneously. Not only offline but also online, when the model is deployed, using the recently introduced *Same-Decision Probability* [3]. This allows us to dynamically minimizes the setup's resource usage, while maintaining good detection accuracy.

This work was triggered by our previous work on monitoring domestic activities based on multi-channel acoustics. The resulting dataset has been consolidated in the publicly available *SINS database* [6, 1]. The data was gathered using an acoustic sensor network where various low-cost microphones were distributed in an home environment. The SINS database is different from many other publicly available datasets as the data is recorded as a continuous stream and activities were being performed in a spontaneous manner. The original paper proposes to compute the Mel-Frequency Cepstral Coefficients (MFCC) as well as its Delta and Acceleration coefficients as input features, and makes use of a Support Vector Machine based classifier. Computing these features and classifier comes with a high computational cost as it requires various transformations of the raw measurements. Executing these operations in an always-on fashion will drain the battery of the targeted embedded sensing nodes, requiring alternative solutions.

Theoretically, operating under scarce resource constraints can be avoided by pushing the computational overhead to a more powerful computing device or to the *cloud*. One can, for example, use the sensor nodes to only collect the data and then communicate the raw measurements towards another device which then performs the necessary computations and returns the result. In practice, however, this approach comes with various concerns as communication of the raw signals is often infeasible for sensors with high sampling rates due to bandwidth limitations and communicating this much data might lead to delays and packet losses. In many cases, the only feasible approach is thus to use the sensor node or embedded device to also process the measurements and to run the classification algorithm locally. As such, in order to improve the device's resource consumption, each of the components in the classification system should be made resource-aware. As the consumption of the sensors themselves are typically negligible in the classification pipeline [18], resource-aware classification implies we should obtain battery savings during feature generation as well as execution of the classifier. As there is no free lunch, battery savings in this context typically require to explore the trade-off space between computational cost and classification accuracy.

To enable this, we propose a two-layered classifier with as top layer a Bayesian Network that allows us to make decisions and as bottom layer a mixture of classifiers operating on features with diverse computational costs (e.g. Tree Augmented Naive Bayes or Convolutional Neural Networks). We rely on Bayesian networks for the decision making, i.e. the top layer, as they come with several advantages: (1) They are generative models and can deal elegantly with missing data. This allows us to dynamically decide which of the bottom-layer classifiers and features to compute or request. (2) They allow us to reason about uncertainty. In the context of activity tracking, for example, this allows us to precisely quantify the probability of transitions between activities. (3) They allow us to reason about the Same-Decision Probability. Same-Decision Probability (SDP) is a recently introduced technique to investigate the usefulness of observing additional features [3]. SDP is the probability one would make the same-decision, had

one known the value of additional features. Our resource-aware method relies on SDP to dynamically evaluate the usefulness of spending additional computational resources.

The main contributions of this paper are thus twofold, both leading to increased resource-awareness and resource-efficiency of an always-on embedded classifier pipeline. Firstly, we present a two-layered classifier that dynamically chooses which features to compute and which models to evaluate. Secondly, we show how same-decision probability can be used in a resource-aware classifier to select features, both in an offline and an online manner. We apply variations of the proposed two-layered model to investigate the computational cost versus classification accuracy trade-off space. We validate our models using both the real-world SINS database and the *Human Activity Recognition Using Smart-phones* (HAR) benchmark dataset [2].

2 Background and Notations

Upper-case letters (Y) denote random variables and lower case letters (y) denote their instantiations. Bold letters represent sets of variables ($\mathbf{Y}$) and their instantiations ($\mathbf{y}$).

2.1 Bayesian Classifiers

A *Bayesian network* is a directed acyclic graph where each node represents a random variable and each edge indicates a direct influence among the variables [15]. The network defines a conditional probability distribution $\Pr(X|\mathbf{Pa}(X))$ for every variable X, where $\mathbf{Pa}(X)$ are the parents of X in the graph. The joint distribution of the network can be factored into a product of conditional distributions as follows:

$$\Pr(X^1, X^2, \ldots, X^n) = \prod_{i=1}^{n} \Pr(X^i|\mathbf{Pa}(X^i))$$

A *Bayesian classifier* is a Bayesian network where one variable (or more) represents the class and each of the features are represented by variables that can be observed. For example, a Tree Augmented Naive Bayes (TAN) classifier has the class variable as its root, and all other variables represent features that have as parents the root and one other feature variable. A TAN classifier comes with less strong assumptions compared to a Naive Bayes classifier and, as a result, typically comes with a higher classification accuracy while still maintaining moderate computational complexity.

A *dynamic Bayesian network* is defined by two networks: B_1, which specifies the prior or initial state distribution $\Pr(\mathbf{Z}_1)$, and $B_{\rightarrow}$, a two-slice temporal BN that specifies the transition model $\Pr(\mathbf{Z}_t|\mathbf{Z}_{t-1})$. Together, they represent the distribution

$$\Pr(\mathbf{Z}_{1:T}) = \Pr(\mathbf{Z}_1) \prod_{t=2}^{T} \Pr(\mathbf{Z}_t|\mathbf{Z}_{t-1})$$

2.2 Bayesian Inference and Knowledge Compilation

Many techniques exist to perform inference in a Bayesian Network. In this paper we use knowledge compilation and weighted model counting [5]. This is a state-of-the-art inference technique for Bayesian networks and Dynamic Bayesian networks [19].

After encoding BNs into a logic knowledge base, we use the c2d knowledge compiler [4] to obtain an Arithmetic Circuit. An Arithmetic Circuit is a directed acyclic graph where leaf nodes represent variables and their weights, and where inner nodes represent additions and multiplication. Inference in the BN now boils down to performing an upward pass through the AC. Hence, the size of the obtained arithmetic circuit directly denotes the required number of computational operations.

2.3 Same-Decision Probability

Recently, the concept of Same-Decision Probability (SDP) has been introduced in the context of Bayesian Networks that are employed to support decision making in domains such as diagnosis and classification [3]. Intuitively, SDP is the probability that we would have made the same threshold-based decision, had we known the state of some hidden variables pertaining to our decision. Within the context of classification, SDP denotes the probability we would classify a data instance towards the same class had we known the value of some additional features.

Let d and $\mathbf{e}$ be instantiations of the decision variable D and evidence variables $\mathbf{E}$. Let T be a threshold. Let $\mathbf{X}$ be a set of variables distinct from D and $\mathbf{E}$. The same-decision probability (SDP) is computed as:

$$SDP_{d,T}(\mathbf{X}|\mathbf{e}) = \sum_{\mathbf{x}} \Big(\underbrace{[\Pr(d|\mathbf{x},\mathbf{e}) =_T \Pr(d|\mathbf{e})]}_{\alpha} \cdot \Pr(\mathbf{x}|\mathbf{e}) \Big) \tag{1}$$

Here, the equality $=_T$ holds if both sides evaluate to a probability on the same side of threshold T, and $[\alpha]$ is 1 when α is true and 0 otherwise. We can compute the SDP on a given Bayesian Network using knowledge compilation and model counting [14].

3 Two-layered model: Building blocks

In order to explore and enhance the accuracy versus computational cost trade-off, we propose to use a two-layered model of classifiers (see Figure 1). Intuitively, the bottom-layer consists of various state-of-the-art classifiers and their classification result is combined by means of a top-layer network. Additionally, feature generation comes with a certain cost and, in many applications, this feature extraction cost even significantly surpasses the cost of classifier model evaluations. Hence, also this component should be taken into account in the accuracy versus resource trade-off.

To summarize, resource-aware classification provides three degrees of freedom that we will explore: (1) Generation of the required features; (2) Training and evaluation of the bottom-layer classifiers; (3) A top-layer network that dynamically combines the bottom-layer classifiers.

3.1 Feature Generation

Sensor nodes typically produce a continuous stream of measurements which are sampled at a certain rate. For microphones, as used for the SINS dataset, the sampling rate

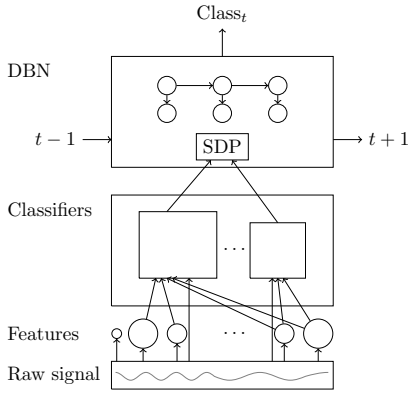


Fig. 1: Overview of the two-layered classifier. Arrows indicate flow of information.

is 16 kHz while for the accelerometer and gyroscope, as used for the HAR dataset, the sampling rate is 50 Hz. Optionally, the measurements can be preprocessed with, for example, a noise filter. Next, one applies a fixed-width sliding window with a certain overlap to split the measurements into sets with a fixed number of values. Finally, these fixed-width windows of values can be used to compute various features.

While we do not aim to formally quantize this in general, we differentiate between features based on their computational cost of generating these features. We report the cost of the features used in the empirical evaluation in Sec. 5 and more detailed analysis about the cost of various features is available in other research [10, 8, 12]. We differentiate between three general types:

1. Raw signal: Not all classifiers require features. Convolutional Neural Networks, for example, can be trained on the raw signal as they perform implicit feature extraction in the convolutional layers of the network. This type of feature preprocessing does not require any effort and have thus no cost.

2. Cheap features: We refer to *cheap* features as those features for which the computational cost is low. For example, computing the *min*, *max* and *mean* of a window is rather cheap as it only requires a constant number of operations for each of the values in the window. Hence, resource-aware feature generation and classification have a preference for this type of features.

3. Expensive features: We refer to *expensive* features as those features for which the computational cost is high. For example, computing the Mel-Frequency Cepstral Coefficients of an audio signal is rather expensive as it requires multiple transformations. Hence, considerable battery savings could be obtained in case we can minimize the need to compute these features.

3.2 Bottom-layer: Mixture of Classifiers

Any type of classifier can be used in the bottom-layer, for example (Tree Augmented) Naive Bayes or Convolutional Neural Networks. We only make a distinction how they represent the classes to predict: as a one-vs-all classifier or a multi-class classifier. The

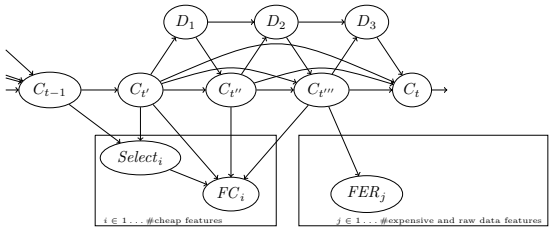


Fig. 2: Overview of our DBN approach.

former typically results in a set of rather small classifiers while the latter yields one large classifier.

A ONE-VS-ALL classifier is an ensemble of classifiers, each trained for one of the N available classes. In other words, N binary classifiers denoting whether a data instance belongs to a particular class or whether it belongs to any of the other classes. The final prediction needs to be derived from all predictions (e.g. using scoring). The advantage is that each classifier can focus on one class and each class is optimally represented. The disadvantage is the potentially large number of classifiers.

In our experimental section, we will use Tree Augmented Naive Bayes (TAN) classifiers for our one-vs-all classifiers as we observed good results but this classifier can be substituted with any other type of classifier.

We choose to only use cheap features to train the one-vs-all classifiers. As a result, the computational cost to evaluate whether a new data instance belongs to a certain class is rather low, at the expense of a potential accuracy impact. This property will be especially exploited in the models we propose in section 4.6 and 4.7 and allows us to push the trade-off space towards models with a lower computational cost.

A MULTI-CLASS classifier represents all classes explicitly in one model. In other words, one classifier directly predicts the particular class a data instance belongs to. The advantage is that only one model needs to be executed. The disadvantage is that not all classes are equally represented.

In our experiments, we will use Tree Augmented Naive Bayes (TAN) classifiers and Convolutional Neural Networks (CNN) for our multi-class classifiers but both of them could be substituted by other types of classifiers depending on the application.

We choose to use all available features, i.e. cheap and expensive features to train the TAN multi-class classifier. The CNN, on the other hand, operates on the raw data and typically induces a large computational cost due to its large model size. Hence, our multi-class classifiers completely ignore the computational cost while they try to maximize the classification accuracy, allowing us to push the trade-off space towards models with a higher accuracy.

3.3 Top-layer: Dynamic Bayesian Network

The one-vs-all and the multi-class bottom-layer classifiers introduced in the previous sections make use of the available data, i.e. features or raw data, to classify a new data

instance. The top-layer method, on the other hand, aims to dynamically combine the bottom-layer classifiers and reasons about their output.

As top-layer method we use a dynamic Bayesian network (DBN). Compared to a static network, a DBN additionally takes into account information from the past to reason about the present state. Within the context of monitoring and activity tracking, this leverages information that the activity a person was performing in the past will be of great help to classify the current activity: (1) There is quite a high probability the person remains doing the same activity; (2) The transition of one activity towards particular other activities might be unlikely. For example, if we classify a previous data instance as showering, then in the next time step this person is most likely still showering or drying herself. The transition of showering towards another activity is much less likely.

We present the general strategy we followed to build the DBN (see Fig. 2). The concrete various instantiations between which we compare are detailed in Sec. 4. The dynamic Bayesian Network observes the previous state (S_{t-1}), i.e. the probability distribution over the previous activities, and the result of up to three different types of bottom-layer classifiers (represented by C_t-primes) that use a selection of cheap features (FC), possibly together with expensive features and raw data (FER). This information is combined in an incremental and probabilistic manner to compute the likelihood of being in a state, i.e. the probability distribution of the current activity. Not all classifiers use all cheap features. When moving from C_{t-1} towards $C_{t'''}$ incrementally more features are being computed. This selection is encoded in the $Select_i$ variables.

The method we use in the top-layer comes with some specifications that are fulfilled by a Bayesian approach: (1) We want to dynamically choose which of the bottom-layer classifiers to evaluate. This is coordinated by the D_i variables that based on the certainty of the classifiers C_t-prime(s) decides to query more classifiers/features (i.e. go to a next C_t-prime(s)) or not and pass the current state directly to the next time step (through C_t); (2) We want to quantify the level of uncertainty in the dependencies between activities and classifiers. (3) We want to reason about the usefulness of evaluating additional bottom-layer classifiers by means of the SDP. This mechanism can be used to decide on the outcome of the D variables.

All parameters in the proposed dynamic Bayesian network classifier can be trained in a straightforward manner by counting if the labels are known during training. The transition probabilities result from counting the available transition in the data. The observation probabilities represent the trustworthiness of the classifiers and are obtained by counting the correctly classified data instances from the set of trained bottom-layer classifiers.

4 Two-layered model: Variations

We now propose several models that are variations of the the two-layered models where the components are specified in detail. This allows us to explore and enhance the classification accuracy versus computational resources trade-off space. Each of the models makes use of one or more of the base classifiers introduced in the previous section.

The title of each of the following subsections uses a shorthand description of each of the proposed models, which we will use further on when we refer to these models. On

the right side, we denote a more informative description of how the various classifiers are combined. Arrows indicate a chain of different classifiers. On top of the arrow is indicated whether we use a simple threshold to decide to evaluate the next classifier or the more advanced SDP.

4.1 MC MC

Our first model is a static, baseline approach where we use the multi-class (MC) classifier trained over all the available classes in the dataset, without a dynamic network layer on top of it. Classifying a new data instance with our MC classifier is straightforward, as it directly denotes the most likely class the data instance belongs to.

The **advantage** if this model is that we can use any state-of-the-art method to train the multi-class classifier and, hence, this model should result in state-of-the-art classification accuracies. The **disadvantage** is that this model does not take into account the computational cost of evaluating the classifier, nor does it exploit temporal correlations.

4.2 OVA $\text{OVA} \xrightarrow{\text{NA}} \text{MC}$

Our second static, baseline model firstly relies on one-vs-all (OVA) classifiers to classify a new data instance, without taking into account the prediction of the previous time step. In case of agreement, i.e exactly one of the classifiers predicts that the data instance belongs to a particular class, we pick the corresponding class as the predicted class for the data instance. In case of non-agreement (NA), we evaluate the multi-class classifier.

The **advantage** if this model is that the use of the OVA classifiers significantly reduces the computational cost. The **disadvantage** is that we do not have a way to reason about the outcome of the OVA classifiers and, in worst case, we still have to rely on the MC classifier for quite a lot of data instances, nor does this model exploits temporal correlations.

4.3 DBN_{OVA} DBN_{OVA}

To combine the results of the OVA classifiers in a probabilistic way, we feed them as observations into a dynamic Bayesian network top-layer. Specifically, our DBN has a corresponding variable for each of the available OVA classifiers. The class variable is a multi-valued variable where the number of values is equal to the number of classes in the data. To use this model, we first evaluate each of OVA classifiers and feed their output to the DBN. Next, we compute the posterior probability of the class variable in the DBN and the value with the highest probability denotes the predicted class.

The **advantage** of this model is that the computational cost is further reduced as we completely avoided the use of the multi-class classifier and expensive features. Additionally the dynamic top-layer network allows us to exploit temporal correlations between consecutive data instances. The **disadvantage** is that the classification accuracy is expected to be quite below the state-of-the-art as we solely rely on cheap features.

4.4 $\mathbf{DBN_{MC}}$ $\mathbf{DBN_{MC}}$

We now combine the DBN top-layer with the multi-class classifier where the DBN contains exactly one observed variable corresponding to this MC classifier. Hence, we first evaluate the multi-class classifier and feed its output to the DBN.

The **advantage** of this model is that the DBN takes into account the probability distribution of the previously classified data instance and is expected to increase the classification accuracy compared to solely using the multi-class classifier. The **disadvantage** is that additionally evaluating the DBN classifier increases the computational cost compared to the MC alternative.

4.5 $\mathbf{DBN_{OVA+MC}}$ $\mathbf{DBN_{OVA}} \xrightarrow{\mathbf{TH}} \mathbf{DBN_{OVA+MC}}$

We now combine the previous two models and extend the DBN classifier with the one-vs-all as well as with the multi-class classifiers. Hence, the DBN contains a corresponding variable for each of the available classifiers. In order to reduce the computational cost, we first only compute the cheap features, evaluate each of the OVA classifiers and feed them to the DBN. In case the most likely class predicted by the DBN has a certain probability, i.e. it is above a certain threshold (TH), we proceed by picking this class. In case the most likely class comes with a rather low probability, we do additionally compute the expensive features, evaluate the MC classifier and feed it to the DBN.

The **advantage** of this model is that it nicely allows us to explore the accuracy versus computational cost trade-off space by changing the threshold (TH). A high threshold would require a considerable amount of MC classifier evaluations but most likely also increases the classification accuracy. A low threshold, on the other hand, would result in a lower classification accuracy but also significantly reduces the computational cost. The **disadvantage** is that the use of a threshold is not robust as it ignores how likely it is that the predicted class changes by evaluating additional classifiers

4.6 $\mathbf{SDP_{OFFLINE}}$ $\mathbf{DBN_{OVA\text{-}SDP}} \xrightarrow{\mathbf{TH}} \mathbf{DBN_{OVA}} \xrightarrow{\mathbf{TH}} \mathbf{DBN_{OVA+MC}}$

We can now ask ourselves the question whether it is always useful to evaluate each of the OVA classifiers or to additionally evaluate the MC classifier in case a certain threshold is not reached. Consider the following scenario; Our DBN classifier predicts a data instance to belong to a particular class with a rather high probability. For the next data instance, the single OVA classifier of this particular class predicts the data instance belongs again to that class. Intuitively, there is only a small probability that observing additional classifiers will change the predicted class.

The above reasoning is exactly what we can compute with the SDP. As we will see in our experimental evaluation, the SDP in this case is indeed very high denoting there is a high probability we will still predict the same class, even after observing additional classifiers. Important to note is that, in this case, we can precompute the SDP offline and, hence, we do not need to spend any additional computational resources at run-time.

The **advantage** of this model is that it allows us to further reduce the computational cost as we now start out by only evaluating a single one-vs-all classifier. The **disadvantage** is that we still have to rely on a threshold in case the single one-vs-all classifier predicts the class to be changed.

4.7 $\mathbf{SDP_{ONLINE}}$ $\mathbf{DBN_{OVA\text{-}SDP} \xrightarrow{TH} DBN_{OVA} \xrightarrow{SDP} DBN_{OVA+MC}}$

In the previous model, we computed the SDP in an offline manner. This allowed us to decide whether we do need to observe additional classifiers in case the corresponding OVA classifier predicts a new data instance to belong to the same class as the previous data instance. In this model, we now aim to further optimize the opposite case, where the class of the new data instance is most likely not the same as the previous one.

Specifically, we now use the SDP in an online manner to investigate whether it is useful to additionally evaluate the MC classifier given that we have observed all OVA classifiers. While computing the SDP online comes with a certain cost, it will be less expensive compared to generating complex features in case the cost of the latter surpasses the cost of evaluating the OVA classifiers.

The **advantage** of this model is that it allows us to fully exploit the classification accuracy versus computational cost trade-off space, where we aim to maximally avoid the need to evaluate additional classifiers. The **disadvantage** of this approach is that we should aim to also reduce the number of SDP computations and therefore we again rely on a threshold to decide whether we compute the SDP or not.

5 Experiments

The goal of our experimental evaluation is to: (1) explore the classification accuracy versus computational cost trade-off space with the various models we introduced in the previous section and (2) investigate the use of the Same-Decision Probability for resource-aware classification.

5.1 Datasets

The first dataset (SINS) [6] aims to track the daily activities performed by a person in a home environment, based on the recordings of an acoustic sensor network. The daily activities are performed in a spontaneous manner and are recorded as a continuous acoustic stream. As a result, the dataset contains also the transitions of going from one activity towards the next activity. The data is recorded with a microphone and sampled at a rate of 16 kHz. We make use of the data recorded in the bathroom where the dataset distinguishes seven activities; showering, getting dry, shaving, toothbrushing, absence, vacuum cleaner and others.

The second benchmark dataset, Human Activity Recognition dataset (HAR) [2], aims to recognize the activity a person is performing based on the accelerometer and gyroscope signals measured by a smart phone. The data is sampled at a rate of 50 Hz, preprocessed with various filters, and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap, i.e. 128 readings per window. The raw data as well as 561 precomputed features are available from the authors. The data distinguishes six activities but, in contrast to the SINS dataset, the data is collected by means of a scripted experiment and does not contain the transitions of one activity towards to next activity. We did however concatenate the data in order to introduce such transitions and clearly show the effect of temporal information in a controlled manner.

5.2 Feature Generation

For the SINS data, the expensive features are the mean and standard deviation of the MFCC, the delta, and acceleration coefficients as described in the original paper [6]. These features are computed on a sliding window of 15 seconds with a step size of 10 seconds. For each window, we have 84 complex features. For the cheap features, we first down-sample the original signal to 1.6 kHz and next compute the energy, zero-crossing rate, min and max on windows of 3 seconds within the 15 second window. Hence, for each window we have 20 cheap features.

For the HAR data, the expensive features are the 561 precomputed features available in the dataset computed on the 2.56 sec sliding window. For the multi-class Convolutional Neural Networks classifier we make use of the inertial signals. For the cheap features, we use the mean, standard deviation, max, min and energy for each of the three axes of the time domain of the body acceleration, gravity acceleration and gyroscope signal. Hence, for each window (data instance) we have 45 cheap features.

5.3 Training of the Classifiers

Preparing data for a one-vs-all classifier leads to an unbalanced dataset as the 'other' label is much more common than the current class. We balance the data by duplicating data instances of the minority class and we add a small amount of Gaussian noise (0.05 of the standard deviation).

To train the Tree Augmented Naive Bayes classifiers, we make use of Weka with the default options [7]. To compile the obtained Bayesian Networks into an Arithmetic Circuit, in order to estimate their computational cost, we make use of ACE with the default options.[3] To train the CNN for the HAR data we make use of Keras with the hyper-parameters set as in the corresponding paper [17]. To model the Dynamic Bayesian Network, we make use of Problog where we use the default options for compilation and inference.[4]

For the SINS data we use 7-fold cross validation where each of the folds corresponds to one day of the recorded data. For the HAR data, we randomly generate 7 train and test sets, out of the train and test data provided by the dataset, and report average results.

5.4 Cost Computation

The computational cost of generating the features, evaluating the base classifiers, and computing the SDP in an online manner is shown in Table 1. For the TAN classifiers as well as the DBN we report the number of edges in the compiled Arithmetic Circuit representation. This is, up to a constant, identical to the required number of operations. For the CNN classifier we computed the number of required operations after training the network with the hyper-parameters described in the corresponding paper [17]. For the features we use an approximated number of operations based on our experience when

[3] `http://reasoning.cs.ucla.edu/ace/`
[4] `https://dtai.cs.kuleuven.be/problog/`

implementing these in hardware. For the cheap features, this is a cost of 5 operations for each of the values in the fixed window. For expensive features on the HAR data, we assume a cost of 10 operations for each of the values in the fixed window. For the expensive features on the SINS data, we estimate the number of required operations to be 600k to compute the MFCC, delta and acceleration coefficients per second. Following Equation (1), computing the SDP requires to evaluate the corresponding Bayesian network $2 \cdot N + 1$ times with N the number of classes in the data. In our case, online computations of the SDP is done on the DBN. We observe that the cost of computing the SDP in an online manner is rather low compared to computing the expensive features and evaluating the multi-class classifier.

Table 1: Computational cost (required number of operations) for features generation, evaluation of the base classifiers and online computation of the SDP. We use HAR_{DL} to refer to the HAR dataset with the CNN multi-class classifier

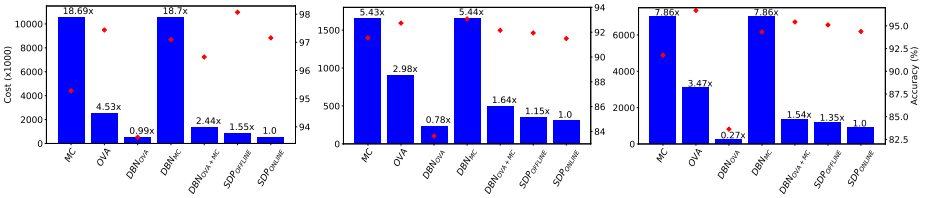
		SINS (x1000)	HAR (x1000)	HAR_{DL} (x1000)
ONE-VS-ALL	Features	500	29	29
	Base Classifier	54	204	204
MULTI-CLASS	Features	10000	690	0
	Base Classifier	581	965	7000
DBN	Base Classifier	4	4	4
SDP (ONLINE)		52	52	52

5.5 Cost versus Accuracy

The total cost and obtained classification accuracy for each of the different models introduced in the previous section is reported in Figure 3. We show the average cost of classifying one data instance and additionally report this cost relative to $\text{SDP}_{\text{OFFLINE}}$. While DBN_{OVA} comes with the lowest cost, as it does not require any of the expensive features nor the multi-class classifier, it also comes with the lowest accuracy. As expected, the MC and DBN_{MC} models have the highest computation cost as they only rely on the expensive features and multi-class classifier. OVA performs slightly better than MC because if the OVA classifiers do not agree, the decision is forwarded to the MC method. Using a DBN model and deciding incrementally what features to query offers a substantial decrease in resource usage (e.g. on SINS, DBN_{OVA+MC} needs half the number of operations of OVA). Applying SDP to make decisions instead of a simple threshold further reduces the number of operations (e.g. on SINS about 35%).

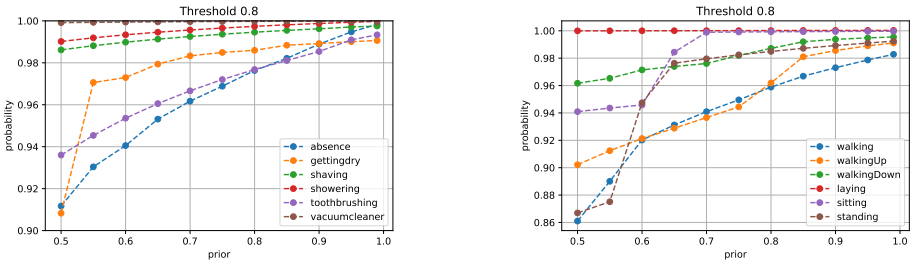
Influence of SDP. The result of the offline SDP computation, as discussed in section 4.6, is depicted in Figure 4. Such offline computation cannot take into account the exact probability distribution of the previously classified data instance. Therefore, we compute the SDP for different prior distributions (x-axis). We observe that for each of the activities, the SDP is rather high, meaning that the probability of classifying the data instance differently after observing additional features, i.e. evaluating additional classifiers, is rather low. This is also observed in Figures 5a, 6a and 7a.

In order to investigate the $\text{SDP}_{\text{ONLINE}}$ method, we additionally introduce an *optimized* baseline model. This optimized model acts as the $\text{SDP}_{\text{OFFLINE}}$ but only picks the



(a) Results for SINS. (b) Results for HAR. (c) Results for HAR with CNNs.

Fig. 3: Cost of the models (bars) versus accuracy (diamonds).



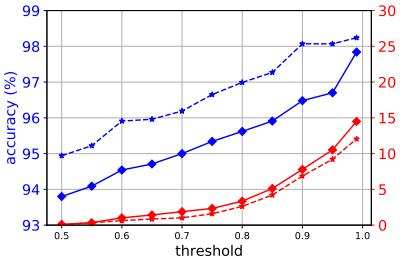
(a) Results for SINS. (b) Results for HAR.

Fig. 4: Offline computation of the SDP with a threshold of 0.8.

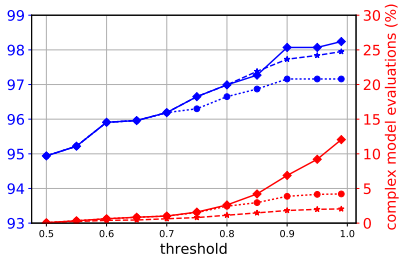
distribution computed after observing the MC classifier if the most likely state differs from the one before observing the MC classifier. This model represents the ideal case if we would be able to look ahead. The result for the different models for different thresholds is depicted in figures 5b, 6b and 7b. While the $\text{SDP}_{\text{ONLINE}}$ method is not able to completely meet with the *optimized* baseline model, it allows us to further trade off accuracy with computational resources.

Influence of the temporal sequence. The HAR dataset consists of separate sequences. To illustrate the effect of the DBN model that learns about sequences we generate three types of concatenated HAR sequences. A *realistic* sequence that is sampled from an expert defined transition probability distribution, a *fixed* sequence that is sampled from a deterministic transition distribution, and a *random* sequence that is sampled from a random transition probability distribution. In each of our other experiments, we made use of the *realistic* sequence.

Figure 8 shows that the highest accuracy is obtained for the *fixed* sequence while, at the same time, it requires the least computational resources. The opposite is observed for the *random* sequence, i.e. lowest accuracy for the highest computational cost. This clearly shows the benefit of the dynamic Bayesian network as it almost completely captures the deterministic transitions while it offers almost no benefit in case the order of activities is random.

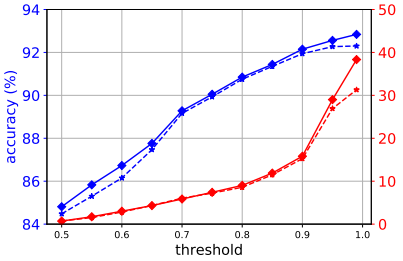


(a) Comparison of $\text{DBN}_{\text{OVA+MC}}$ (solid line) with $\text{SDP}_{\text{OFFLINE}}$ (dashed line).

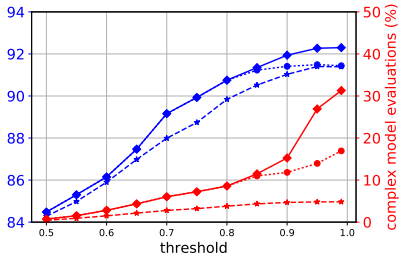


(b) Comparison of $\text{SDP}_{\text{OFFLINE}}$ (solid line) with $\text{SDP}_{\text{ONLINE}}$ (dotted line) and the *optimized* model. (dashed line).

Fig. 5: Results for SINS.



(a) Comparison of $\text{DBN}_{\text{OVA+MC}}$ (solid line) with $\text{SDP}_{\text{OFFLINE}}$ (dashed line).

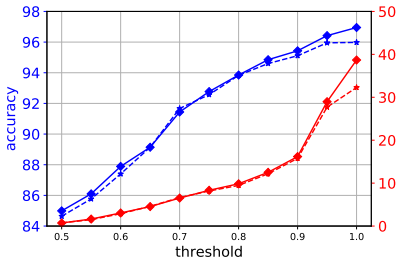


(b) Comparison of $\text{SDP}_{\text{OFFLINE}}$ (solid line) with $\text{SDP}_{\text{ONLINE}}$ (dotted line) and the *optimized* model. (dashed line).

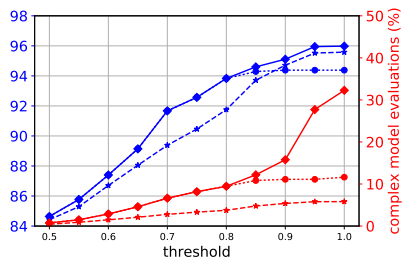
Fig. 6: Results for HAR.

6 Conclusions

We propose a two-layered computational model which enables an enhanced trade-off between computational cost and classification accuracy. The bottom layer consist of various classifiers with each a certain computational cost to generate features and to evaluate the classifier. The top layer consists of a Dynamic Bayesian network that probabilistically combines the bottom layer networks and takes into account the obtained probability distribution of the previous time step to classify the compute the current state. Additionally, the top layer network allows us to dynamically decide which classifiers to evaluate and which features to compute by means of simple threshold based decisions or Same-Decision Probability. The latter is computed in an offline as well as online manner and allows us to further push the computational resources versus classification accuracy trade-off space. We validate our methods on the real-world SINS database, where domestic activities are recorded with an accoustic sensor network, as



(a) Comparison of $\text{DBN}_{\text{OVA+MC}}$ (solid line) with $\text{SDP}_{\text{OFFLINE}}$ (dashed line).



(b) Comparison of $\text{SDP}_{\text{OFFLINE}}$ (solid line) with $\text{SDP}_{\text{ONLINE}}$ (dotted line) and the *optimized* model. (dashed line).

Fig. 7: Results for HAR with CNNs.

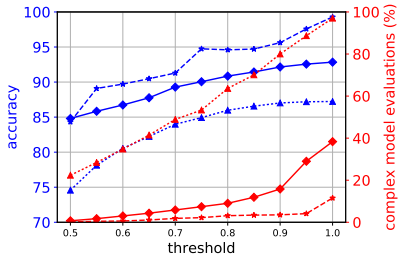


Fig. 8: Results for three different HAR sequences, being a *realistic* sequence (solid line) as also used in our other experiments, a *fixed* sequence (dashed line) and a *random* sequence (dotted line).

well as the Human Activity Recognition (HAR) benchmark dataset and show that resources can be dynamically minimized while maintaining a good accuracy.

Acknowledgements. This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme. Grant agreements 715037 Re-SENSE: Resource-efficient sensing through dynamic attention-scalability and 694980 SYNTH: Synthesising Inductive Data Models.

References

1. DCASE Challenge 2018: Monitoring of domestic activities based on multi-channel acoustics, `http://dcase.community/challenge2018`
2. Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J.L.: A Public Domain Dataset for Human Activity Recognition Using Smartphones. In: 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013
3. Choi, A., Xue, Y., Darwiche, A.: Same-decision probability: A confidence measure for threshold-based decisions. International Journal of Approximate Reasoning **53**(9), 1415–1428 (2012)

4. Darwiche, A.: New advances in compiling CNF to decomposable negation normal form. In: Proceedings of ECAI. pp. 328–332 (2004)
5. Darwiche, A.: Modeling and Reasoning with Bayesian Networks. Cambridge University Press (2009)
6. Dekkers, G., Lauwereins, S., Thoen, B., Adhana, M.W., Brouckxon, H., van Waterschoot, T., Vanrumste, B., Verhelst, M., Karsmakers, P.: The SINS Database for Detection of Daily Activities in a Home Environment Using an Acoustic Sensor Network. In: Proc. of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE) (2017)
7. Frank, E., Hall, M.A., Witten, I.H.: The WEKA Workbench, "Data Mining: Practical Machine Learning Tools and Techniques". Morgan Kaufmann, 4th edn. (2016)
8. Ghasemzadeh, H., Ostadabbas, S., Guenterberg, E., Pantelopoulos, A.: Wireless Medical-Embedded Systems: A Review of Signal-Processing Techniques for Classification. IEEE Sensors Journal **13**, 423–437 (2013)
9. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems **29**(7), 1645–1660 (2013)
10. Kantorov, V., Laptev, I.: Efficient feature extraction, encoding and classification for action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2593–2600 (2014)
11. Korhonen, I., Parkka, J., Van Gils, M.: Health monitoring in the home of the future. IEEE Engineering in medicine and biology magazine **22**(3), 66–73 (2003)
12. Mitra, J., Saha, D.: An Efficient Feature Selection in Classification of Audio Files. arXiv preprint arXiv:1404.1491 (2014)
13. Olascoaga, L.I.G., Meert, W., Bruyninckx, H., Verhelst, M.: Extending Naive Bayes with Precision-tunable Feature Variables for Resource-efficient Sensor Fusion. In: AI-IoT ECAI. pp. 23–30 (2016)
14. Oztok, U., Choi, A., Darwiche, A.: Solving PP PP-Complete Problems Using Knowledge Compilation. In: Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning (2016)
15. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann (1988)
16. Piatkowski, N., Lee, S., Morik, K.: Integer undirected graphical models for resource-constrained systems. Neurocomputing **173**, 9–23 (2016)
17. Ronao, C.A., Cho, S.B.: Human Activity Recognition with Smartphone Sensors Using Deep Learning Neural Networks. Expert Syst. Appl. **59**(C), 235–244 (Oct 2016)
18. Shnayder, V., Hempstead, M., Chen, B.r., Allen, G.W., Welsh, M.: Simulating the power consumption of large-scale sensor network applications. In: Proceedings of the 2nd international conference on Embedded networked sensor systems. pp. 188–200. ACM (2004)
19. Vlasselaer, J., Meert, W., Van den Broeck, G., De Raedt, L.: Exploiting local and repeated structure in dynamic Bayesian networks. Artificial Intelligence **232**, 43–53 (2016)
20. Xu, Z.E., Kusner, M.J., Weinberger, K.Q., Chen, M., Chapelle, O.: Classifier cascades and trees for minimizing feature evaluation cost. Journal of Machine Learning Research **15**(1), 2113–2144 (2014)