Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Efficient probability intervals for classification using inductive venn predictors

Dimitrios Boursinos*, Xenofon Koutsoukos

Institute for Software Integrated Systems, Vanderbilt University, Nashville, 37235, USA

A R T I C L E   I N F O

Article history:
Received 30 December 2022
Revised 1 April 2023
Accepted 29 May 2023
Available online 5 June 2023

Keywords:
Deep neural networks
Assurance monitoring
Inductive Venn predictors
Probability intervals

A B S T R A C T

Learning enabled components are frequently used by autonomous systems and it is common for deep neural networks to be integrated in such systems for their ability to learn complex, non-linear data patterns and make accurate predictions in dynamic environments. However, their large number of parameters and their use as black boxes introduce risks as the confidence in each prediction is unknown and output values like softmax scores are not usually well-calibrated. Different frameworks have been proposed to compute accurate confidence measures along with the predictions but at the same time introduce a number of limitations like execution time overhead or inability to be used with high-dimensional data. In this paper, we use the Inductive Venn Predictors framework for computing probability intervals regarding the correctness of each prediction in real-time. We propose taxonomies based on distance metric learning to compute informative probability intervals in applications involving high-dimensional inputs. By assigning pseudo-labels to unlabeled input data during system deployment we further improve the efficiency of the computed probability intervals. Empirical evaluation on image classification and botnet attacks detection in Internet-of-Things (IoT) applications demonstrates improved accuracy and calibration. The proposed method is computationally efficient, and therefore, can be used in real-time. The code is available at https://github.com/dboursinos/Efficient-Probability-Intervals-Classification-Inductive-Venn-Predictors.

1. Introduction

Modern Deep Neural Network (DNN) architectures have the capacity to be trained using high-dimensional data and make accurate predictions in dynamic and uncertain environments. This ability makes them a common choice for many autonomous system applications. However, when DNNs are used as black boxes in safety-critical systems, they may result in disastrous consequences if it is not possible to reason about their predictions.

The training of a Learning Enabled Component (LEC) requires specification of the task, performance measure for evaluating how well the task is performed, and experience in the form of training and testing data. An LEC, such as a DNN, during system operation exhibits some nonzero error rate and the true error rate is unknown and can only be approximated during design time using the available data. The problem is that the approximations are not always good. Confidence values, such as the softmax probabilities which are used by most DNNs for classification, are usually greater than the actual posterior probability that the prediction is correct. Important factors that make modern DNNs overconfident are the depth, width, and techniques like weight decay, and batch normalization [1].

Our objective is to complement the predictions made by DNNs with a computation of confidence. The confidence can be expressed as probability intervals that characterize the correctness of the DNN prediction. This is an upper and a lower bound that asymptotically contain the probability of correctness of the prediction. An efficient and robust approach must ensure that the actual accuracy of a DNN is contained in the computed intervals and the width of the intervals is small. We focus on computationally efficient algorithms that can be used in real-time. The proposed approach is based on the Inductive Venn Predictors (IVP) framework [2]. IVP computes the probability intervals for an unknown input leveraging knowledge it has acquired from previous predictions on labeled data. IVPs are defined by a taxonomy which splits data into categories according to their similarity. Well-calibrated multi-probabilistic predictions are generated by computing the class distribution of labeled data in each category. Most of the IVP or Venn Predictors (VP) applications in the literature are evaluated on low-dimensional data. More specifically the datasets that have been used for experimental evaluation contain: 16x16 grayscale images in [2,3], tabular data with 50 and 57 attributes

* Corresponding author.
E-mail addresses: dimitrios.boursinos@vanderbilt.edu (D. Boursinos), xenofon.koutsoukos@vanderbilt.edu (X. Koutsoukos).

https://doi.org/10.1016/j.patcog.2023.109734
0031-3203/© 2023 Elsevier Ltd. All rights reserved.

in [4], tabular data with 8 to 13 attributes in [5], tabular data with 6 and 12 attributes in [6], tabular data with 5 to 18 attributes in [7]. In this paper we use distance metric learning methods to transform high-dimensional data into lower-dimensional embedding representations so that IVP can by applied in applications with high-dimensional data, like images.

In our previous work [8], we measured the likelihood of a prediction based on p-values using the Inductive Conformal Prediction (ICP) framework. Even though p-values can be used to produce prediction sets with a well-calibrated error-rate bound, their interpretation is less direct than that of probabilities and it is harder to reason about their meaning. Moreover many times they are confused with probabilities. However the difference is significant. According to [8], a set predictor is formed according to the p-values associated with each possible class and the probability of error is less or equal to a chosen significance level. The problem with ICP is that the guarantees for the set predictors only apply a priori. IVPs, on the other hand, are used to compute probability intervals that express the posterior conditional probability of a label given the training set and test input.

The estimation of reliable predictive uncertainty has become an important part of many modern machine learning components used in safety-critical applications. Even though many of the proposed methods produce well-calibrated models, their application in the real world is challenging. In [9,10], new training algorithms and loss functions are proposed to achieve well-calibrated DNNs. These approaches require training DNN models from scratch and cannot be used with pre-trained ones. Another category of calibration methods like the Platt's scaling [11] and temperature scaling [1] proposes ways of post-processing the outputs of already trained models to produce calibrated confidence measures. In [12,13], it is shown that these methods are not as well-calibrated as it is reported especially when the validation data are not independent and identically distributed (IID) and in the presence of distribution shifts. The Conformal Prediction (CP) framework is developed to compute prediction sets to satisfy a desired significance level [14]. The confidence value assigned to each possible class is in the form of *p*-values which is less intuitive than estimating the confidence as probabilities. Another way of obtaining confidence information about predictions is by using algorithms based on the Bayesian framework. The use of this framework, however, require some prior knowledge about the distribution generating the data. In the real world, this distribution is unknown and it has to be chosen arbitrarily. In [15], it is shown that the predictive regions produced by Gaussian Processes, a popular Bayesian machine learning approach, may be incorrect and misleading when the correct prior is not known.

The main contribution of our work is the use of dynamic categories that increase in size with new data during runtime. The second contribution is the computation of low-dimensional, appropriate, embedding representations of the original inputs in a space where the Euclidean distance is a measure of similarity between the original inputs, in order to handle high-dimensional inputs in real-time. The third contribution is the implementation of four different taxonomies that split the low-dimensional data into categories based on their similarity. Then, we implement categories that can increase in size to include unlabeled data during runtime in real-time as they are encountered. ICP is used to compute candidate labels for the unlabeled data and they are placed to their corresponding categories according to the chosen taxonomy. Last, we present an empirical evaluation of the approach using two datasets for image classification problems with a large number of classes as well as detection of botnet attacks in an IoT device. The underlying models are chosen according to the input size and shape while satisfying the low-latency and low-power requirements, to meet the resource constraints of the variety of use cases [16]. This paper presents an empirical study with three datasets as a proof-of-concept for improving the efficiency of IVPs. Our exhaustive comparison of all proposed and baseline taxonomies indicates that the taxonomy choice affects the quality of the computed probability intervals. Our experiments show that the taxonomy decision is driven by our ability to compute embedding representations of data points that form tight clusters. The method presented in this paper has higher accuracy on test set predictions compared to our previous work in [8].

2. Problem formulation

A perception component in an autonomous system aims to observe and interpret the environment in order to provide information for decision-making. When machine learning models are used for decision making, it is essential for them to provide metrics related to their uncertainty. For example, a perception component can be used for classifying traffic signs in autonomous vehicles. The problem is to complement the decisions of the perception component with a computation of confidence. An efficient and robust approach must ensure a small and well-calibrated error rate to enable real-time operation. An accurate estimation of a relatively small error-rate, according to the specification, can maximize the autonomous operation while limiting the number of inputs for which an accurate prediction cannot be made.

During system operation, for each new input a prediction is made, usually by a LEC and the objective is to compute a valid measure of the prediction's confidence. The objective is twofold: (1) provide guarantees for the error rate of the prediction and (2) limit the number of input examples for which a confident prediction cannot be made. Well-calibrated confidence in terms of probabilities can be used for decision-making, for example, by generating warnings when human intervention is required. To improve properties like calibration and efficiency, the assurance monitoring and classification system needs to take into account and adapt to data observed during runtime in real-time without suspending the operation.

The VP framework can produce predictions with confidence intervals that guarantee to include the true probabilities for each class output to occur [2]. The confidence intervals for a test input are generated by considering the class distribution of labeled inputs assigned to the same category that are collected offline and are available to the system. In the literature, VP implementations use Support Vector Machines (SVMs) or DNN classifiers to create categories of labeled data [2,4,7]. The additional problem we are considering is the computation of appropriate embedding representations that can lead to more efficient VPs. Using such representations taxonomies need to be be defined to form categories of similar input data in a lower-dimensional space. This, not only reduces the memory requirements, but is also more efficient in producing more informative intervals. The efficiency and calibration of IVP improves as categories contain more data. The accuracy of the predictions as well as the efficiency and calibration of the probability intervals are affected by the availability of labeled data. Out of two probability intervals, more efficient, or informative, is considered the one with the lowest uncertainty regarding the true accuracy. A common problem for classification components that make use of ML models is the acquisition of appropriate labeled data. To address the problem of limited availability of labeled data, pseudo-labels need to be assigned to the unlabeled data that can be classified confidently. During execution time, input data arrive one by one. Their conformity with the training set needs to be evaluated and using statistical methods reject the labels that are less likely to be true for the newly received, unlabeled, input data.

3. Distance metric learning

As pointed in [17], ML models perform well when they need to make decisions on test inputs that are similar to the labeled data used for training. Furthermore, in [18] it is shown that areas in the feature space that lack training data lead to much higher error-rate. When the data are expressed in high-dimensions, the analysis of the feature space as well as the similarity estimation between two data points is not straightforward. This problem commonly appears in content-based image retrieval (CBIR) systems [19]. The metric chosen to define the distance between images can strongly affect the performance and computing distance metrics in high-dimensions is computationally expensive and requires a large memory capacity to store all available data in their original form. The efficiency can be improved by using distance metric learning to map the original data to lower-dimensional representations on an embedding space where the distance can be computed efficiently.

A siamese network is composed using two copies of the same neural network with shared parameters [20] as shown in Fig. 1. During training, each identical copy of the siamese network is fed with different training samples x_1 and x_2 belonging to classes y_1 and y_2. The embedding representations produced by each network copy are $r_1 = \text{Net}(x_1)$ and $r_2 = \text{Net}(x_2)$. The learning goal is to minimize the Euclidean distance between the embedding representations of inputs belonging to the same class and maximize it for inputs belonging to different classes as described below:

$$\begin{cases} \min d(r_1, r_2), & \text{if } y_1 = y_2 \\ \max d(r_1, r_2), & \text{otherwise} \end{cases} \tag{1}$$

This optimization problem can be solved using the contrastive loss function [21]:

$$L(r_1, r_2, y) = y \cdot d(r_1, r_2) + (1 - y) \max[0, m - d(r_1, r_2)]$$

where y is a binary flag equal to 0 if $y_1 = y_2$ and to 1 if $y_1 \neq y_2$ and m is a margin parameter. In particular, when $y_1 \neq y_2$, $L = 0$ when $d(r_1, r_2) \geq m$, otherwise the parameters of the network are updated to produce more distant representations for those two elements. The margin parameter is used so that only hard example pairs, close to each other, will be used for training.

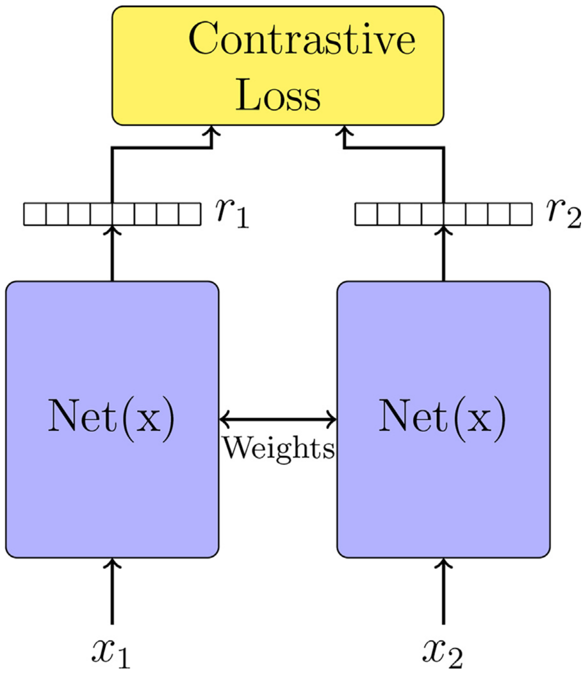

Fig. 1. Siamese network structure.

We denote $f: X \rightarrow V$ the mapping from the input space X to the embedding space V by a single copy of the DNN pair in the siamese network. Using the trained network, the embeddings $v_i = f(x_i)$ are computed and stored for all the training data x_i. The same transformation takes place online as new test input data arrive to the system.

4. Siamese-based IVP

Venn Predictors is a machine learning framework that can be combined with existing classifier architectures for producing well-calibrated multi-probability predictions under the IID assumption [2,22]. This means that the confidence assigned to a prediction is a probability distribution which in effect defines lower and upper bounds regarding the probability of correctness for all possible classes. VPs are well-calibrated and the probability bounds asymptotically contain the corresponding true conditional probabilities (proof in [2]). However the framework is computationally inefficient as it requires training the underlying algorithm after every new test input. Computational efficiency can be addressed using the Inductive Venn Predictors [4,6], an extension of the VP framework.

Central to the VP and IVP frameworks is the definition of a Venn taxonomy. This is a way of clustering data points into a number of categories according to their similarity and is based on an underlying algorithm. For example a taxonomy can be defined to put in the same category examples that are classified in the same class by a DNN. The main idea of our approach is that the taxonomy can be defined efficiently by learning embedding representations of the inputs for which the Euclidean distance is a measure of similarity. To compute the embedding representations of the inputs we train a siamese network using contrastive loss as shown in Section 3.

We consider the training examples, $z_1, \ldots, z_l$ from $\boldsymbol{Z}$, where each z_i is a pair (x_i, y_i) with x_i the feature vector and y_i the corresponding label. We also consider a test input x_{l+1} which we wish to classify. The available training examples are split into two parts: the proper training set with q examples and the *calibration set* with $l - q$ examples. IVP assumes that the calibration and test sets are independent and identically distributed (IID) generated from the same but usually unknown probability distribution. The examples in the proper training set are used to train the siamese network which is used to define different Venn taxonomies. The roll of the taxonomy is to divide the $l - q$ calibration examples into a number of categories based on their similarity. This process takes place during the design time.

As proved in [2] the probabilistic outputs assigned to each class by the VP are well-calibrated regardless of the choice of the Venn taxonomy and this holds in practice for IVP as well [4]. However, the choice of the taxonomy affects the efficiency of the IVP. The probability intervals are desirable to be relatively narrow to minimize the uncertainty in the probability of correctness as well as create better separation between the probabilities of each class. In [23] we proposed four different Venn taxonomies based on distance metric learning that we briefly present here for completeness. The first two taxonomies are based on a k-Nearest Neighbors classifier. The naive approach, that we call *k-NN* V_1, trains a k-NN classifier using the embedding representations of the proper training set. Then the calibration data, as well as each new test input, are placed to a category that is defined by the k-NN prediction using the computed embedding representations. That is, for a data point x_{l+1} that needs to be placed into a category, its embedding representation is computed using the siamese network, $r_{l+1} = f(x_{l+1})$ and its k nearest training data are found. Depending on the class $\hat{y}_{l+1}$ that most neighbors belong to, the data point is assigned to the category

$$k_{l+1} = \hat{y}_{l+1}. \tag{2}$$

This taxonomy creates a number of categories that is equal to the number of classes in the dataset. Then, we extended this taxonomy to more accurately split the data into categories by taking into account how many of the k nearest training data points are labeled different than the predicted class. For a data point x_{l+1} with embedding representation r_{l+1} that needs to be placed into a category we compute the k-nearest neighbors in the training set and store their labels in a multi-set Ω. We call this taxonomy *k-NN* V_2 and the category where x_{l+1} is placed is computed as:

$$k_{l+1} = \hat{y}_{l+1} \times \left(k - \left\lfloor \frac{k}{c} \right\rfloor\right) + |i \in \Omega : i \neq \hat{y}_{l+1}| \tag{3}$$

where $\hat{y}_{l+1}$ is the k-NN classification of r_{l+1}, k is the number of nearest neighbors and c is the number of different classes. This taxonomy aims at further improving the similarity of the data in each category leveraging the classifier's confidence. It is expected that the more similar labeled neighbor training data points, the higher the chance of the corresponding class being the correct one. That way each category of *k-NN* V_1 is further split into $k - \left\lfloor \frac{k}{c} \right\rfloor$ new categories.

By utilizing the ability of siamese networks to form clusters of similar data we can further reduce the Venn taxonomy computational requirements when there is a large amount of training data. Each class cluster i corresponding to class Y_i, $i = 1\ldots, c$ can then be represented by its centroid $\mu_i = \frac{\sum_{j=1}^{n_i} r_j^i}{n_i}$, where r_j^i is the embedding representation of the j^{th} training example from class Y_i and n_i is the number of training examples labeled as Y_i. We propose another family of taxonomies based on the nearest centroids. The *NC* V_1 places the calibration data as well as each new test input to a category that is the same as the class assigned to their nearest centroid. The category, where an example x_{l+1} is placed, is computed as:

$$k_{l+1} = \arg\min_{j=1,\ldots,c} d(r_{l+1}, \mu_j) \tag{4}$$

where d the Euclidean distance. This leads to a number of categories that is equal to the number of classes in the dataset. An extension of this taxonomy, the *NC* V_2, attempts to form more accurate categories by taking into account the classification confidence. We expect data points of the same class to be more similar to each other when their embedding representations are placed at similar distances to their class centroid. That way each category of *NC* V_1 is further split into two categories based on how close an example x_{l+1} is to its nearest centroid:

$$k_{l+1} = 2 \times \arg\min_{j=1,\ldots,c} d(r_{l+1}, \mu_j) + h, \tag{5}$$

$$h = \begin{cases} 0, & \text{if } d(r_{l+1}, \mu_{\min}) \leq \theta \\ 1, & \text{otherwise} \end{cases}$$

where $\mu_{\min} = \arg\min_{j=1,\ldots,c} d(r_{l+1}, \mu_j)$ is the distance to the nearest centroid and θ a chosen distance threshold.

After placing the calibration data into categories using the underlying algorithm for the taxonomy, during execution time we consider a test input x_{l+1} and place it in a category k_{l+1}. The true class y_{l+1} is unknown so all possible classes Y_j are considered as candidates one after the other. The empirical probability assigned to each candidate class is:

$$p(Y_j) = \frac{|\{(x^*, y^*) \in k_{l+1} : y^* = Y_j\}|}{|k_{l+1|}}. \tag{6}$$

k_{l+1} will always be non-empty as it will contain at least the new example x_{l+1}. This creates a probability distribution for the label y_{l+1} computed as the ratio of data belonging to each class in a category. That way we can compute the maximum and minimum probabilities assigned to each class Y_j. When the true class is assumed to be Y_j then the count of examples labeled as Y_j in k_{l+1} will increase by one and result in the maximum probability assigned to class Y_j, $U(Y_j)$. For all the other classes Y_i, $i = 1, \ldots, c : i \neq j$ the computed probability will be their minimum probability $L(Y_j)$. These are the two bounds that define the probability intervals $[L(Y_j), U(Y_j)]$ for each class. The predicted class for the classification is computed as:

$$j_{best} = \arg\max_{j=1,\ldots,c} \overline{p(Y_j)} \tag{7}$$

where $\overline{p(Y_j)}$ is the mean of the probability interval assigned to Y_j. Along with the class $Y_{j_{best}}$ the IVP framework outputs the probability interval $[L(Y_{j_{best}}), U(Y_{j_{best}})]$. By temporarily placing the new example to each of the n categories, one at a time, we compute a set of probability distributions that compose the multi-probability prediction of the IVP, $P_{l+1}^{k_i} = \{p^{k_i}(Y_j) : k_i \in \{k1, \ldots, k_n\}, Y_j \in \{Y_1, \ldots, Y_c\}\}$. That way the initial probability intervals assigned to each class for each category as well as the class classification can be computed offline using the labeled calibration data. The steps taking place during execution are illustrated in Fig. 2.

5. Inductive venn predictors with dynamic categories

Categories with more data lead to computation of probability intervals that are narrower and better calibrated. The categories are commonly formed during design time using labeled data that were not used for training and remain unchanged during execution. However, many times the available data during design time are not enough to form categories that satisfy specifications regarding the probability intervals. Our method utilizes dynamic categories that expand in size during runtime by including newly encountered data with pseudo-labels. For the pseudo-labeling we use the ICP framework for its error-rate guarantees it provides. ICP approaches the labeling as a hypothesis testing problem that rejects the labels that are less likely to be correct. Given a test input x_{l+1}, ICP computes a prediction set Γ^ϵ of labels with enough evidence to be the true label, where ϵ the significance level of the hypothesis testing. Hypothesis testing is a statistical method used to make decisions on whether a hypothesis is true based on a finite number of data. The null hypothesis, H_0, is the argument believed to be true and the *alternative hypothesis*, H_1, is the argument to be proven true based on the collected data. We determine whether to accept or reject the alternative hypothesis based on the likelihood of the null hypothesis being true, given by p-values. We are certain that exactly one of the labels in Y is true so $\hat{y}_{l+1} = y_{l+1}$ is the null hypothesis. This hypothesis needs to be rejected for the $c - 1$ incorrect labels so $\hat{y}_{l+1} \neq y_{l+1}$ is the alternative hypothesis. ICP computes prediction sets Γ^ϵ such that $P(y_{l+1} \notin \Gamma^\epsilon) < \epsilon$, for any choice of ϵ with the underlying assumption that all examples (x_i, y_i), $i = 1, 2, \ldots$ are IID generated from the same but typically unknown probability distribution and exchangeable [24].

We utilize the error-rate bound guarantees of ICP to update the IVP categories in real-time during system execution. The process is shown in Fig. 3. The new unlabeled input x is first transformed to a low-dimensional embedding representation v. According to the chosen taxonomy, the representation v corresponds to one of the predefined categories. The prediction $\hat{y}$ as well as the probability interval $[L(\hat{y}), U(\hat{y})]$ is computed using the IVP framework and the class distribution in the assigned category. After

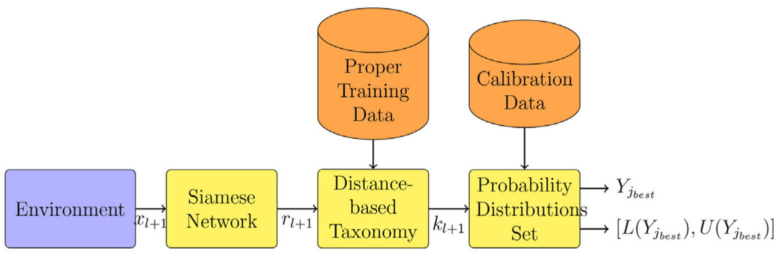

Fig. 2. IVP classifier based on distance metric learning.

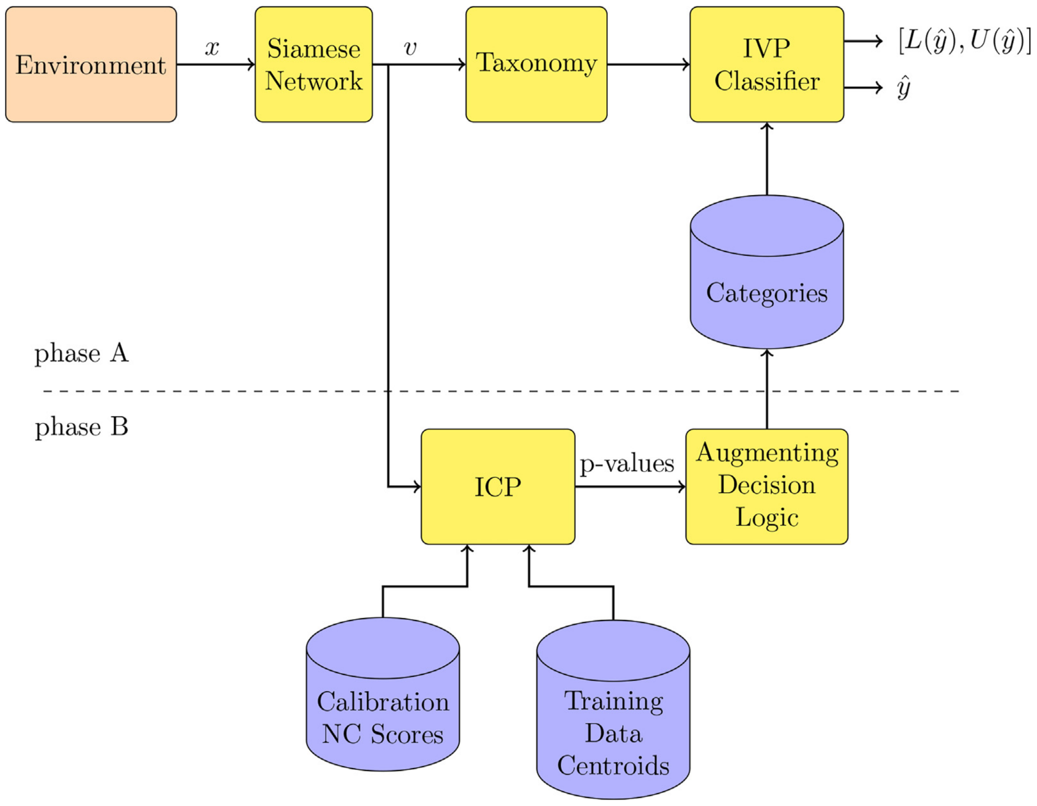

Fig. 3. Execution time workflow.

the prediction output, in phase B, we evaluate if we can pseudo-label the input x and add it to the assigned category pool. We use ICP to compute the set Γ^{ϵ} of candidate labels that show evidence of being correct. The class distribution in the category computed by the IVP taxonomy is updated to include the labels in Γ^{ϵ}.

Central to the application of ICP is a nonconformity function or nonconformity measure (NCM) which shows how different a labeled input is from the examples in the training set. The same siamese network that is trained for IVP is used to define the ICP NCMs. The proper training set and the calibration set used for ICP are the same that are used to train the siamese network and assign data to categories for the IVP framework. In [25], we presented different NCMs based on low-dimensional embedding representation. According to the application a particular NCM may be more suitable. The k-NN NCM finds the k most similar examples of a test input x in the training data and counts how many of those are labeled different than the candidate label y. We denote $f: X \rightarrow V$ the mapping from the input space X to the embedding space V defined by either a siamese or a triplet network. Using the trained neural network, the encodings $v_i = f(x_i)$ are computed and stored for all the training data x_i. Given a test input x with encoding $v = f(x)$, we compute the k-nearest neighbors in V and store their labels in a multi-set Ω. The k-NN NCM of input x with a candidate label y is defined as

$$\alpha(x, y) = |i \in \Omega : i \neq y|. \tag{8}$$

The Nearest Centroid NCM simplifies the task of computing individual training examples that are similar to a test example when there is a large amount of training data. We expect examples that belong to a particular class to be similar to each other so for each class y_i we compute its centroid $\mu_{y_i} = \frac{\sum_{j=1}^{n_i} v_j^i}{n_i}$, where v_j^i is the embedding representation of the j^{th} training example from class y_i and n_i is the number of training examples in class y_i. The nonconformity function is defined as:

$$\alpha(x, y) = \frac{d(\mu_y, v)}{\min_{i=1,\ldots,n: y_i \neq y} d(\mu_{y_i}, v)} \tag{9}$$

where $v = f(x)$. It should be noted that for computing the nearest centroid NCM, we need to store only the centroid for each class.

The advantage of the nearest centroid NCM, in Eq. 9 is that it requires minimal memory and computational power as for the NC scores to be computed, only the centroids of the training data need to be stored. This is more significant in relatively large datasets. However, this methods only works well when our distance learning methods can create tight clusters around the centroids. If this is not possible because of the dataset complexity we can use the k-NN NCM, in Eq. 8. This function does not assume tight clustering of data belonging to the same class around a single cluster. Data with different characteristics may belong in the same class and distance metric learning methods can produce multiple clusters for a given class. This function, on the other hand, requires the whole training set to be stored and more computational power to compute the k-NN of a given test input in the training set.

The NC scores give us an indication of which classes appear to conform better with an input. However their values can range depending on the dataset and it is hard to set thresholds and choose the pseudo-labels for unlabeled inputs. ICP normalizes the NC scores and translates it into p-values using a calibration dataset (X^c, Y^c). The nonconformity scores of the calibration data are computed in design time and stored in A, where:

$$A = \{\alpha(x, y) : (x, y) \in (X^c, Y^c)\}. \tag{10}$$

For a test example with feature vector x and a candidate prediction j, the nonconformity can be computed similarly to the calibration examples. The empirical p-value for each candidate label j is:

$$p_j(x) = \frac{|\{\alpha \in A : \alpha \geq \alpha(x, j)\}|}{|A|}. \tag{11}$$

Then, a set prediction Γ^ϵ for the input x can be computed as the set of all labels j such that $p_j(x) > \epsilon$. The category κ that x is assigned to is computed by the IVP taxonomy of choice and if $|\Gamma^\epsilon| \geq 1$, the class distribution in κ is updated to include the labels in Γ^ϵ.

6. Evaluation metrics

The performance of IVP based on the proposed taxonomies is evaluated regarding the accuracy, calibration and efficiency. The objective is for the computed probability intervals to contain the true probability of correctness for each prediction. The probability interval for a given input x with predicted class $\hat{y}$ is $[L(\hat{y}), U(\hat{y})]$. Equivalently, the probability that $\hat{y}$ is not the correct classification will be in the complimentary interval $[1 - U(\hat{y}), 1 - L(\hat{y})]$, called error probability interval. The true probability of correctness for a single prediction is unknown so the correctness of the computed intervals is evaluated over a number of samples. To do this we use the following metrics:

- cumulative errors

$$E_n = \sum_{i=1}^{n} err_i, \tag{12}$$

$$err_i = \begin{cases} 1, & \text{if classification } \hat{y}_i \text{ is incorrect} \\ 0, & \text{otherwise} \end{cases}$$

- cumulative lower and upper error probabilities

$$LEP_n = \sum_{i=1}^{n} [1 - U(\hat{y}_i)], \qquad UEP_n = \sum_{i=1}^{n} [1 - L(\hat{y}_i)] \tag{13}$$

To compare the IVP based on our proposed taxonomies with the baseline taxonomies, scalar metrics are used that represent the performance regarding accuracy, calibration, and efficiency. Unlike the NN classifiers that produce a single softmax probability for each class, the IVP framework produces probability intervals. For the computation of the evaluation metrics the probability assigned to a class Y_j will be $\overline{p(Y_j)}$ like in Eq. 7. The accuracy of an IVP implementation is evaluated as the number of correct classifications over the number of attempted classifications and it is computed as

$$accuracy = 1 - \frac{E_n}{n}. \tag{14}$$

An efficient, or informative, IVP is one that makes predictions with small diameter probability intervals and their median is as close to zero or one. The most popular quality metrics for probability assessments are the negative log-likelihood (NLL) and the Brier score (BS) [26]. NLL is the simplest out of the two and only considers the probability assigned to the predicted class in Eq. 7. It is computed as

$$NLL = -\sum_{i=1}^{n} \sum_{j=1}^{c} t_i^j \log(o_i^j), \tag{15}$$

where $o_i^j = \overline{p(Y_j)}$ of example i and t_i^j the one-hot representation of the ground truth classification label y_i of example i, that is

$$t_i^j = \begin{cases} 1, & \text{if classification } y_i = Y_j \\ 0, & \text{otherwise} \end{cases}$$

This metric is minimized by producing intervals that are narrow and have median probability close to one assigned to the correct class. Computational issues may occur as the log score explodes if we observe an event that the classifier considers impossible. BS is computed as

$$BS = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} (o_i^j - t_i^j)^2 \tag{16}$$

This is, in effect, the mean squared error of the predictions. Unlike NLL, BS considers the probabilities assigned to all possible classes and will penalize probability intervals assigned to incorrect classes that are not close to zero. There are different views in the literature regarding which scoring rule is more appropriate[27]. emphasizes in the importance of the locality property, meaning, the scoring rule should only depend on the probability of events that actually occur and only NLL satisfies this. On the other hand, [28] states that a scoring rule should be symmetric and only BS satisfies this. This means that if the true class probability is p and the predicted probability is $\hat{p}$, then the score should be equal to the case where the true probability is $\hat{p}$ and the predicted probability is p. However, we think that both metrics produce useful insights in probability assessment so both are reported in our experiment results. The interval size has a significant role on how informative and interpretable a prediction is. Narrower probability intervals are more informative than wider ones because they provide more accurate information about the probability of correctness of a prediction as all possible values of the probability of correctness are tightly clustered around a specific value. We evaluate the size of the probability intervals by computing the average interval diameter as

$$D = \frac{\sum_{i=1}^{n} U(\hat{y}_i) - \sum_{i=1}^{n} L(\hat{y}_i)}{n} \tag{17}$$

A well-calibrated IVP computes probability intervals that are representative of the true correctness likelihood. Formally a model is well-calibrated when

$$\mathbb{P}\big(\hat{y} = Y \,|\, \hat{p} = p\big) = p, \qquad \forall p \in [0, 1] \tag{18}$$

However, $\hat{p}$ is a continuous random variable so the probability in Eq. 18 cannot be approximated using finitely many samples. According to (18) a measure of miscalibration can be expressed as $\mathop{\mathbb{E}}_{\hat{p}}\Big[\big|\mathbb{P}\big(\hat{y} = y | \hat{p} = p\big) - p\big|\Big]$. The Expected Calibration Error (ECE) [29] computes an approximation of this expected value across bins:

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |\mathrm{acc}(B_m) - \mathrm{conf}(B_m)| \tag{19}$$

where $|B_m|$ is the number of samples in bin B_m, n is the total number of samples and $\mathrm{acc}(B_m)$ and $\mathrm{conf}(B_m)$ are the accuracy and confidence of bin B_m respectively as defined in [29]. Many times in safety critical applications it is more useful to compute the maximum miscalibration of a model than the mean value. This metric is called Maximum Calibration Error (MCE) [29] and is computed as:

$$\mathrm{MCE} = \max_{m\in\{1,\ldots,M\}} |\mathrm{acc}(B_m) - \mathrm{conf}(B_m)| \tag{20}$$

7. Evaluation

In this section, we evaluate the IVPs that use distance-based taxonomies with regard to accuracy, calibration, and efficiency. Additionally, for the evaluation of our proposed taxonomies, we use metrics regarding the performance of the siamese network in clustering similar input data, the execution time of the framework, and the required memory. Then we evaluate our implementation of dynamic taxonomies and compare them with their static distance-based taxonomies counterparts.

7.1. Experimental setup

The embedding representation computations, part of our proposed taxonomies, are not application-specific and can improve the performance of IVP in cases where inputs are high-dimensional. We evaluate the performance of IVP with distance-learning in two different classification problems. First, we have two case studies in image classification. The German Traffic Sign Recognition Benchmark (GTSRB) dataset is a collection of traffic sign images to be classified in 43 classes [30]. The labeled sign images are of various sizes between 15x15 to 250x250 pixels depending on the observed distance. We convert all the images to a fixed shape of 96x96 pixels. The second dataset is the Fruits360 [31]. This dataset contains images of 131 different kinds of fruits and vegetables. The input data are used in their original size, 100x100 pixels.

The second classification problem we consider is the detection of botnet attacks in IoT devices. As part of the evaluation in [32], authors made available data regarding network traffic while infecting different common IoT devices two families of botnets. Mirai and BASHLITE are two common IoT-based botnets and their harmful capabilities are presented in [33]. In the dataset there are data for the following ten attacks:

- BASHLITE Attacks
  1. Scan: Scanning the network for vulnerable devices
  2. Junk: Sending spam data
  3. UDP: UDP flooding
  4. TCP: TCP flooding
  5. COMBO: Sending spam data and opening a connection to a specified IP address and port
- Mirai Attacks
  1. Scan: Scanning the network for vulnerable devices
  2. Ack: Ack flooding
  3. Syn: Syn flooding
  4. UDP: UDP flooding
  5. UDPplain: UDP flooding with fewer options, optimized for higher PPS

Including the benign network traffic, we approach this as a classification problem with eleven classes. The available data are in the form of 115 statistical features extracted from the raw network traffic. The same 23 features, presented in [32], are extracted from five time windows of the most recent 100ms, 500ms, 1.5sec, 10sec, 1min. The features summarize the traffic in each of these time windows that has (1) the same source IP address, (2) the same source IP and MAC address, (3) been sent between the source and destination IP address, (4) been sent between the source and destination TCP/UDP sockets. These features are computed incrementally and in real-time.

The available data are used throughout the evaluation process the same way in every dataset. 10% of the data are taken out to be used for testing and the rest is the training set. The training set is then split into the proper training set and the calibration set. The proper training set is randomly chosen as 80% of the training set and is used to train the underlying models and for the computation of the categories. The calibration set is the remaining 20% of the available training data and it is used only to form the categories during the design time. The reported evaluation results are computed on the separate test set. All the experiments run in a desktop computer equipped with and Intel(R) Core(TM) i9-9900K CPU and 32 GB RAM and a Geforce RTX 2080 GPU with 8 GB memory.

7.2. Baseline taxonomies

To understand the effect of the distance metric learning in IVP we compare it with approaches that use DNN classifiers as underlying algorithms. A variety of Venn taxonomy definitions based on DNNs is proposed in [7]. V_1 assigns two examples to the same category if their maximum softmax outputs correspond to the same class. V_2, divides the examples in the categories defined by V_1 into two smaller categories based on the value of their maximum softmax output. Their chosen threshold for the maximum output to create the two smaller categories is 0.75. V_3 divides the examples in the categories defined by V_1 into two smaller categories but this time based on the second highest softmax output. Their chosen threshold for the second-highest output is 0.25. V_4 divides each category of taxonomy V_1 in two, based on the difference between the highest and second-highest softmax outputs. The threshold for this difference is 0.5. In the same paper, they proposed a fifth taxonomy that creates the categories based on which classes have softmax outputs above a certain threshold. This taxonomy creates 2^C number of categories making its use infeasible in our evaluation datasets.

7.3. Evaluation results

The difficulty to assign an input to a category and the memory demands increase as the size and complexity of the inputs increases. Our goal is to evaluate our method using general-purpose and lightweight DNNs. For the image classification problems, we use the MobileNet [16] architecture for both the embedding representation computation as well as the classifier used for the baseline taxonomies for its low latency and low memory requirements. The trade-off between accuracy and latency is configured by the hyperparameter α. We set $\alpha = 0.5$ in the case of GTSRB and $\alpha = 1$ for the Fruit360. In both cases the embedding representation vectors are of size 128. In the case of the botnet attacks detection, the input data are arranged in vectors of 115 values so we use a fully connected DNN with two hidden layers, the first has 10 units, and the second which produces the embedding representations has 32 units.

After training the siamese network and before it is used as part of the taxonomies we need to evaluate how well it performs in clustering similar inputs. For comparison, we use the embedding space produced by the penultimate layer of the DNN classifier [34]. A commonly used metric of the separation between class clusters is the silhouette coefficient [35]. This metric evaluates how close together samples from the same class are, and far from samples of different classes and takes values in [-1,1]. The results on the silhouette analysis for the test inputs from both datasets are shown

Table 1
Silhouette coefficient comparison.

	Classifier Embeddings	Siamese Embeddings
GTSRB	0.56	0.98
Fruits360	0.52	0.85
Ecobee Thermostat	0.27	0.46

in Table 1. The siamese network produces representations that are well clustered based on their similarity and better than the representations produced by the classifier DNN. This is important for constructing efficient categories using our proposed distance-based taxonomies.

The evaluation results are shown in Table 2. For all datasets, we observe that using the proposed distance-based taxonomies, IVP produces more accurate classifications. Even though the baseline V_1 taxonomy produces probability intervals that are as narrow as the intervals produced by some of the proposed taxonomies, the proposed taxonomies produce better quality intervals by keeping the intervals assigned to the correct class close to 1 and the intervals of the incorrect classes close to 0, as shown by the NLL and BS metrics. The differences in ECE are not significant but most of the proposed taxonomies produce probabilities that are better calibrated in the whole probability space [0,1] with no areas of miscalibration as indicated by MCE.

For illustration, the cumulative upper and lower error probabilities as well as the cumulative error are plotted on the same axis in Fig. 4 for all the taxonomies used in our comparison and test data from the GTSRB dataset. The left side has the resulted plots using distance-based taxonomies and the right side has the resulted plots using the baseline taxonomies. Visually, it is important to focus on three different areas in these plots: the relationship between the cumulative errors shown as a dashed line and the shaded area of the probability intervals, the width of the shaded probability intervals, the number of cumulative errors over time. The computed probability intervals successfully bound the true error-rate for all the presented taxonomies. However, the probability intervals are narrower in most cases when the distance-based taxonomies are used, with the exception of the NC V_2 taxonomy, as shown in the plots in the left side of Fig. 4. Moreover, the cumulative errors over time, when IVPs are used to make predictions is lower when the distance-based taxonomies are used.

The times required for the computation of a classification and the probability intervals when a new input arrives are similar in both the baseline and our proposed taxonomies and indicate they can be used for real-time operation. The speed bottleneck is the computations by the DNNs for either the classifications or the representation mapping. The k-NN computation step in the low-dimensional embedding representation space adds minimal overhead in the execution time. The memory requirements have two main parts: the memory required to store the DNN weights and the memory required to store the categories after calibration. The proposed taxonomies have the additional requirement to store either the embedding representations of the training data to be used by the k-NN or the centroid of each class. The representations of the training data are stored in a $k-d$ tree [36] for fast k-NN computation. With the use of low-dimensional representations, the additional memory required for the nearest centroid based taxonomies is small compared to the underlying DNN size.

7.4. Dynamic categories

IVP implementations typically form the categories during design time using labeled data and the categories remain the same during execution. In Section 5, we introduced a paradigm of updating the categories during execution using unlabeled data in order to further improve the computed probability intervals. In this section we apply this method in CPS test cases to understand how it affects the performance and it is evaluated using the same evaluation metrics presented in Section 6. The taxonomies based on the siamese network we introduced earlier showed better overall performance than other DNN-based taxonomies and will be the baseline in the evaluation of IVP with dynamic categories. The same distance metric based taxonomies are used for both IVP applications. This means that, when a new unlabeled test input arrives, both IVP with static categories and IVP with dynamic categories compute the probability intervals and the classification the same way. When the categories are dynamic, after the initial classification, ICP is used to compute the confidence of each label being correct in the form of p-values and append the current categories by

Table 2
Evaluation metrics results.

Dataset	Taxonomy	Accuracy	NLL	BS	D	ECE	MCE	Time	Memory
GTSRB	V_1	0.994	111.835	0.013	0.009	0.005	0.005	3.6ms	11.2MB
	V_2	0.992	58.104	0.055	0.014	0.011	0.583	6.6ms	11.2MB
	V_3	0.993	75.394	0.038	0.012	0.008	0.750	3.7ms	11.2MB
	V_4	0.991	70.279	0.053	0.013	0.009	0.750	2.9ms	11.2MB
	k-nn V_1	0.998	41.575	0.005	0.009	0.004	0.004	3.2ms	19MB
	k-nn V_2	0.998	41.126	0.005	0.009	0.004	0.004	3.6ms	19.8MB
	NC V_1	0.998	41.575	0.005	0.009	0.004	0.004	2.9ms	3.9MB
	NC V_2	0.996	38.444	0.046	0.017	0.007	0.500	3ms	3.9MB
Fruits360	V_1	0.983	1089.938	0.043	0.019	0.008	0.113	4.4ms	41MB
	V_2	0.986	816.893	0.144	0.025	0.013	0.407	4ms	41.2MB
	V_3	0.985	870.470	0.154	0.025	0.012	0.392	4.6ms	41.2MB
	V_4	0.985	836.295	0.159	0.025	0.013	0.384	2.7ms	41.2MB
	k-nn V_1	0.993	532.314	0.025	0.019	0.010	0.073	3.3ms	127.5MB
	k-nn V_2	0.993	466.311	0.088	0.022	0.011	0.243	3.7ms	128.1MB
	NC V_1	0.991	605.087	0.027	0.019	0.010	0.045	3.6ms	14MB
	NC V_2	0.988	725.556	0.208	0.035	0.018	0.500	3.5ms	14.2MB
Ecobee Thermostat	V_1	0.823	4732.483	0.218	3.8e-04	0.003	0.009	0.7ms	52.2kB
	V_2	0.830	4310.008	0.200	6.1e-04	0.003	0.014	0.7ms	53.2kB
	V_3	0.830	4311.460	0.200	6.2e-04	0.002	0.015	0.7ms	53.2kB
	V_4	0.830	4306.791	0.200	6.1e-04	0.003	0.040	0.6ms	53.2kB
	k-nn V_1	0.935	2872.725	0.113	4.2e-04	0.001	0.003	1.9ms	43.8MB
	k-nn V_2	0.935	2299.023	0.096	19.3e-04	0.006	0.375	2.4ms	43.8MB
	NC V_1	0.794	5550.013	0.255	4e-04	0.006	0.017	1ms	24kB
	NC V_2	0.794	5541.171	0.255	5.8e-04	0.006	0.023	0.9ms	25kB

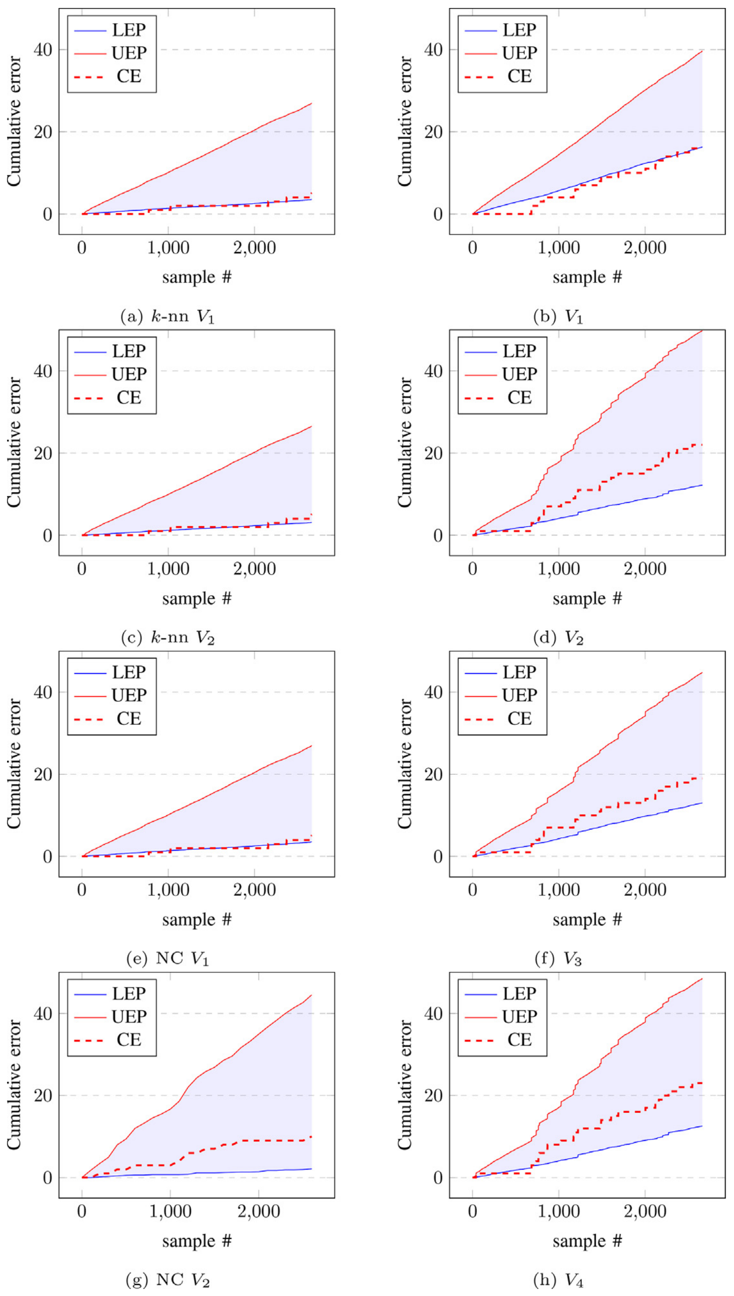

Fig. 4. Cumulative error intervals comparison between our taxonomies and the literature baselines on the GTSRB dataset.

including the confident labels. This extra step introduces an important design time parameter. There are many NCMs for ICP with different trade-offs according to the application. Table 1 presents the silhouette coefficient comparison for the embeddings computed by the siamese network for different datasets and can help us decide the most appropriate NCM for each application. In the case of GTSRB, the siamese network which is used for IVP taxonomies as well as ICP NCMs, forms very clear clusters unlike the Ecobee Thermostat security dataset. This means that the training data in the GTSRB dataset can be replaced by their class centroids and use the nearest centroid NCM for its lower computational requirements. On the other hand, this is not possible for the botnet attacks detection on the Ecobee Thermostat because the clusters are not as dense. In the later case, we use the k-NN NCM which is less sensitive to sparser clustering.

The effects of dynamically appending the categories during execution is shown in Fig. 5–8. The plots show the cumulative lower

Algorithm 1 – Training and Calibration.

```
Require: Training data $(X, Y)$, calibration data $(X^c, Y^c)$
Require: DNN architecture $f$ for distance metric learning based on the siamese network
Require: Taxonomy from Eqs. 2, 3, 4, 5
1: Train $f$ using $(x, y) \in (X, Y)$                    ▷ Training
2: // Compute the representations
3: $V = f(X)$
4: $V^c = f(X^c)$
5: for each $(v_i^c, y_i^c)$ in $(V^c, Y^c), i = 1.l - m$ do
6:     Compute the assigned category $k_i$ using the chosen taxonomy    ▷ Calibration
7:     Add $y_i^c$ to $k_i$
8: end for
9: Store the resulted class distribution of each category
```

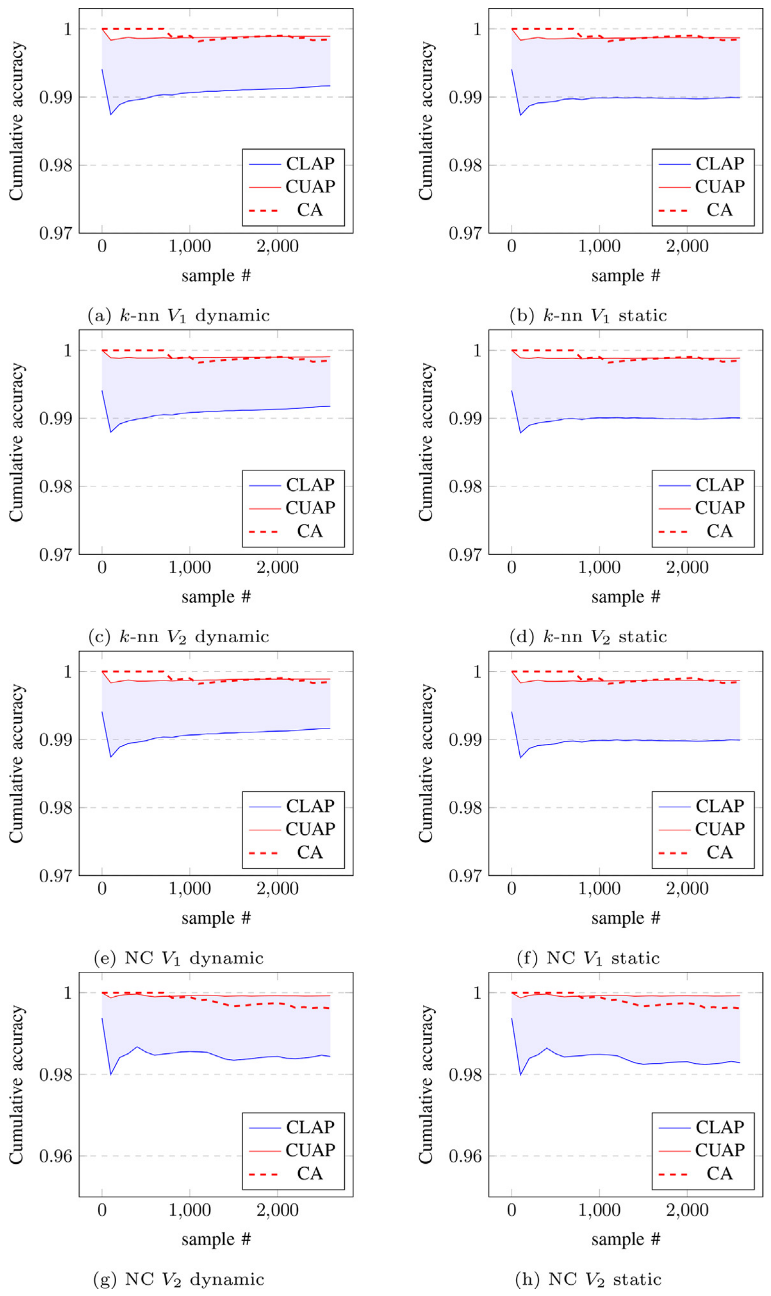

Fig. 5. Cumulative probability intervals comparison between the dynamic and static taxonomies on the GTSRB dataset.

Algorithm 2 – Assurance Monitoring and Classification with Static Categories.

```
Require: Taxonomy from Eqs. 2, 3, 4, 5
Require: Class distribution of each category
Require: Trained siamese network $f$ for distance metric learning
Require: Test input $x_{l+1}$
1: Compute the assigned category $k_{l+1}$ using the chosen taxonomy
2: for each label $j \in 1.n$ do
3:     $L(Y_j) = \frac{|\{(x^*, y^*) \in k_{l+1} : y^* = Y_j\}|}{|k_{l+1}| + 1}$
4:     $U(Y_j) = \frac{|\{(x^*, y^*) \in k_{l+1} : y^* = Y_j\}| + 1}{|k_{l+1}| + 1}$
5: end for
6: Classify $x_{l+1}$ into $j_{best} = \arg\max_{j=1,\dots,c} \overline{p(Y_j)}$
7: Return probability interval $[L(Y_{j_{best}}), U(Y_{j_{best}})]$
```

and upper bounds computed over time during execution using the unlabeled test data from the GTSRB dataset. Furthermore we compute the cumulative accuracy over time by considering the ground truth labels of the test data. All four distance-based taxonomies lead to lower and upper bounds computation that asymptotically bound the true cumulative accuracy, as shown in [2]. Both the static and dynamic IVP classifiers are deployed with the same categories formed by the calibration data with each of the respective taxonomies. As the classifier considers more test data, the initial categories, in the dynamic cases, grow in size. This, not only leads to probability intervals that remain valid, but they get narrower over time compared to their static counterparts.

The results for both CPS datasets are shown in Table 3. Each of the metrics used for evaluation is explained in Section 6. As we noted earlier, the nearest centroid-based taxonomies are not a good choice for the botnet attack detection dataset as the clusters produced by the siamese network do not have clear centroids.

Table 3
Evaluation metrics results.

Dataset	Taxonomy	Accuracy	NLL	BS	D	ECE	MCE
GTSRB	k-nn V_1 static	0.998	41.575	0.005	0.009	0.004	0.004
	k-nn V_1 dynamic	0.998	40.130	0.005	0.007	0.003	0.003
	k-nn V_2 static	0.998	41.126	0.005	0.009	0.004	0.004
	k-nn V_2 dynamic	0.998	39.728	0.005	0.007	0.003	0.003
	NC V_1 static	0.998	41.575	0.005	0.009	0.004	0.004
	NC V_1 dynamic	0.998	40.130	0.005	0.007	0.003	0.003
	NC V_2 static	0.996	38.444	0.046	0.017	0.007	0.500
	NC V_2 dynamic	0.996	36.339	0.046	0.015	0.006	0.500
Ecobee Thermostat	k-nn V_1 static	0.935	2872.725	0.113	4.2e-04	0.0014	0.003
	k-nn V_1 dynamic	0.935	2873.090	0.113	3.7e-04	0.0013	0.001
	k-nn V_2 static	0.935	2299.023	0.096	19.3e-04	0.0058	0.375
	k-nn V_2 dynamic	0.935	2296.795	0.096	18.6e-04	0.0052	0.375
	NC V_1 static	0.794	5550.013	0.255	4e-04	0.006	0.017
	NC V_1 dynamic	0.794	5592.070	0.257	3.5e-04	0.021	0.059
	NC V_2 static	0.794	5541.171	0.255	5.8e-04	0.006	0.023
	NC V_2 dynamic	0.794	5582.567	0.257	5e-04	0.020	0.059

Algorithm 3 – Assurance Monitoring and Classification with Dynamic Categories.

```
Require: Taxonomy from Eqs. 2, 3, 4, 5
Require: Nonconformity function $\alpha$
Require: Calibration nonconformity scores $A$
Require: Significance level threshold $\epsilon$
Require: Class distribution of each category
Require: Trained siamese network $f$ for distance metric learning
Require: Test input $x_{l+1}$
 1: Compute the embedding representation $v_{l+1} = f(x_{l+1})$
 2: Compute the assigned category $k_{l+1}$ using the chosen taxonomy
 3: for each label $j \in 1.n$ do
 4:     $L(Y_j) = \frac{|\{(x^*, y^*) \in k_{l+1} : y^* = Y_j\}|}{|k_{l+1}| + 1}$
 5:     $U(Y_j) = \frac{|\{(x^*, y^*) \in k_{l+1} : y^* = Y_j\}| + 1}{|k_{l+1}| + 1}$
 6: end for
 7: Classify $x_{l+1}$ into $j_{best} = \arg\max_{j=1,\dots,c} \overline{p(Y_j)}$
 8: Return probability interval $[L(Y_{j_{best}}), U(Y_{j_{best}})]$
 9: for each label $j \in 1.n$ do
10:     Compute the nonconformity score $\alpha(x_{l+1}, j)$
11:     $p_j(x) = \frac{|\{\alpha \in A : \alpha \geq \alpha(x_{l+1}, j)\}|}{|A|}$
12:     if $p_j(z) \geq \epsilon$ then
13:         $|\{(x^*, y^*) \in k_{l+1} : y^* = Y_j\}| = |\{(x^*, y^*) \in k_{l+1} : y^* = Y_j\}| + 1$
14:         $|k_{l+1}| = |k_{l+1}| + 1$
15:     end if
16: end for
```

So, in this particular application, we focus on the results produced by the k-NN based taxonomies because of their more ideal performance. We still report the NC based taxonomies results for completeness. The extension to dynamic categories does not have any effect in the classification accuracy. The class j_{best} that an input is classified to is chosen using Eq. 7. The new data with pseudo-labels that were added to populate the existing categories during execution do not change the class distribution in each category enough to change the IVP classification for any given category assignment. The effects of our method is more obvious in the efficiency and calibration related metrics. The number of data in each category affect the width of the computed probability intervals as more data produce narrower, more informative, probability intervals. This is clear for any choice of taxonomy in both datasets. The NLL also improves in the case of GTSRB but has a less obvious change in the botnet attacks detection dataset. The last, and most important observation from these results is in the calibration related metrics. By pseudo-labeling and extending the categories during execution, IVP not only computes narrower probability intervals, but these intervals are better calibrated for all taxonomies in both datasets as indicated by the ECE and MCE metrics.

The use of ICP for pseudo-labeling new unlabeled inputs and appending the existing classes add a minimal overhead of around 0.01ms. This happens because of the efficient computation of the k-NN but also the re-use of the already computed embedding representation of each test input in the initial classification phase. Since the ICP framework that is used to integrate new data in the existing categories shares the same siamese network and embedding representations of the training data that are used for IVP, there is no memory overhead.

8. Conclusion

We presented computationally efficient algorithms based on appropriate embedding representations learned by siamese networks, as a proof-of-concept, that make it possible for IVP to be used with high-dimensional data for real-time applications. Then, we extended these algorithms to utilize unlabeled test inputs gathered during execution to further improve in efficiency and calibration. The evaluation results demonstrate that the IVP framework using distance-based taxonomies produces high accuracy and probability intervals that are efficient and contain the true number of cumulative errors after a number of input samples. The computed probability intervals get narrower and get better calibrated over time when unlabeled test inputs are utilized in the computations. Our choice of lightweight DNNs and small embedding representation size make the approach computationally efficient and can be used in real-time.

There are limitations that can make the application of the presented method on certain datasets challenging. Our workflow uses siamese networks to compute distance metrics. This method can have issues related to scalability. Distance metric learning can become computationally expensive when dealing with large datasets, as it requires computing pairwise distances between a large number of data points. Moreover, distance metric learning can be sensitive to outliers in the dataset, which can distort the learned distance function and lead to poor clustering performance. These limitations affect the computation of the Venn categories as well as the NCMs as both of them rely on the computation of similarity between data points. Clusters that are not tight limit the taxonomy choices to the ones based on k-nn that are more computationally demanding. Finally, limitations of ICP affect the quality of the dynamic Venn categories. Since ICP is designed to guarantee an error rate, the set predictors may be conservative and return prediction sets with multiple classes. Another reason for large prediction sets is when the test data are significantly different than the training distribution resulting in high NC scores. Large prediction sets cause dissimilar unlabeled data to be placed in the same Venn categories affecting the quality of the multi-probabilistic outputs.

Some potential directions for future work include the application of the presented method to more datasets and evaluate the results statistically. Furthermore, the proposed approach can leverage new methods for clustering and pseudo-labeling [37] to improve the performance of dynamic categories. Finally, extend the presented method to work in regression problems using conformal predictive distributions [38] to overcome the limitation of Venn predictors that only work in classification problems.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgment

This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) through contract number FA8750-18-C-0089. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA.

References

[1] C. Guo, G. Pleiss, Y. Sun, K.Q. Weinberger, On calibration of modern neural networks, in: Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17, JMLR.org, 2017, pp. 1321–1330. http://dl.acm.org/citation.cfm?id=3305381.3305518
[2] V. Vovk, A. Gammerman, G. Shafer, Algorithmic Learning in a Random World, Springer-Verlag, Berlin, Heidelberg, 2005.
[3] V. Vovk, G. Shafer, I. Nouretdinov, Self-calibrating probability forecasting, in: NIPS, 2003, pp. 1133–1140.
[4] A. Lambrou, I. Nouretdinov, H. Papadopoulos, Inductive venn prediction, Ann Math Artif Intell 74 (1–2) (2015) 181–201.
[5] H. Papadopoulos, H. Haralambous, Reliable prediction intervals with regression neural networks, Neural Networks 24 (8) (2011) 842–851.
[6] A. Lambrou, H. Papadopoulos, I. Nouretdinov, A. Gammerman, Reliable probability estimates based on support vector machines for large multiclass datasets, in: IFIP International Conference on Artificial Intelligence Applications and Innovations, Springer, 2012, pp. 182–191.
[7] H. Papadopoulos, Reliable probabilistic classification with neural networks, Neurocomputing 107 (2013) 59–68.
[8] D. Boursinos, X. Koutsoukos, Assurance monitoring of learning-enabled cyber-physical systems using inductive conformal prediction based on distance learning, artificial intelligence for engineering design, Analysis and Manufacturing 35 (2) (2021) 251–264, doi:10.1017/S089006042100010X.
[9] G. Pereyra, G. Tucker, J. Chorowski, L. Kaiser, G. Hinton, Regularizing neural networks by penalizing confident output distributions, arXiv preprint arXiv:1701.06548 (2017).
[10] A. Kumar, S. Sarawagi, U. Jain, Trainable calibration measures for neural networks from kernel mean embeddings, in: J. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, Vol. 80 of Proceedings of Machine Learning Research, PMLR, Stockholmsmässan, Stockholm Sweden, 2018, pp. 2805–2814.
[11] J.C. Platt, Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, in: Advances in Large-Margin Classifiers, MIT Press, 1999, pp. 61–74.
[12] A. Kumar, P.S. Liang, T. Ma, Verified uncertainty calibration, in: Advances in Neural Information Processing Systems, 2019, pp. 3792–3803.
[13] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, J. Snoek, Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift, in: Advances in Neural Information Processing Systems, 2019, pp. 13991–14002.
[14] G. Shafer, V. Vovk, A tutorial on conformal prediction, J. Mach. Learn. Res. 9 (2008) 371–421. http://dl.acm.org/citation.cfm?id=1390681.1390693
[15] H. Papadopoulos, V. Vovk, A. Gammerman, Regression conformal prediction with nearest neighbours, J. Artif. Int. Res. 40 (1) (2011) 815–840.
[16] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural net- works for mobile vision applications, arXiv:1704.04861 (2017).
[17] G. Marcus, Deep learning: a critical appraisal, arXiv preprint arXiv:1801.00631 (2018).
[18] G.M. Weiss, Learning with rare cases and small disjuncts, in: In Proc. of Twelfth International Conference on Machine Learning, Morgan Kaufmann, 1995, pp. 558–565.
[19] V.N. Gudivada, V.V. Raghavan, Content based image retrieval systems, Computer (Long Beach Calif) 28 (9) (1995) 18–22.
[20] G. Koch, R. Zemel, R. Salakhutdinov, Siamese neural networks for one-shot image recognition, ICML Deep Learning Workshop, Vol. 2, Lille, 2015.
[21] I. Melekhov, J. Kannala, E. Rahtu, Siamese network features for image matching, in: 2016 23rd International Conference on Pattern Recognition (ICPR), IEEE, 2016, pp. 378–383.
[22] V. Balasubramanian, S.-S. Ho, V. Vovk, Conformal prediction for reliable machine learning: theory, adaptations and applications, 1st, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2014.
[23] D. Boursinos, X. Koutsoukos, Reliable probability intervals for classification using inductive venn predictors based on distance learning, in: 2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS), 2021, pp. 1–7, doi:10.1109/COINS51742.2021.9524144.
[24] G. Shafer, V. Vovk, A tutorial on conformal prediction, Journal of Machine Learning Research 9 (3) (2008).
[25] D. Boursinos, X. Koutsoukos, Assurance monitoring of cyber-physical sys- tems with machine learning components, in: Tools and Methods of Competitive Engineering, 2020, pp. 27–38.
[26] G.W. Brier, Verification of forecasts expressed in terms of probability, Mon. Weather Rev. 78 (1) (1950) 1–3, doi:10.1175/1520-0493(1950)078<0001:VOFEIT>2.0.CO;2.
[27] R. Benedetti, Scoring rules for forecast verification, Mon. Weather Rev. 138 (1) (2010) 203–211.
[28] R. Selten, Axiomatic characterization of the quadratic scoring rule, Experimental Economics 1 (1) (1998) 43–61.
[29] M.P. Naeini, G. Cooper, M. Hauskrecht, Obtaining well calibrated prob- abilities using bayesian binning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 29, 2015.
[30] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, Man vs. computer: benchmarking machine learning algorithms for traffic sign recognition, Neural Networks 32 (2012) 323–332, doi:10.1016/j.neunet.2012.02.016. Selected Papers from IJCNN 2011
[31] H. Mureşan, M. Oltean, Fruit recognition from images using deep learning, Acta Universitatis Sapientiae, Informatica 10 (1) (2018) 26–42, doi:10.2478/ausi-2018-0002.
[32] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, Y. Elovici, N-Baiot—network-based detection of iot botnet attacks using deep autoencoders, IEEE Pervasive Comput. 17 (3) (2018) 12–22.
[33] C. Kolias, G. Kambourakis, A. Stavrou, J. Voas, DDoS in the IoT: mirai and other botnets, Computer (Long Beach Calif) 50 (7) (2017) 80–84, doi:10.1109/MC.2017.201.
[34] G.E. Hinton, Learning multiple layers of representation, Trends Cogn. Sci. (Regul. Ed.) 11 (10) (2007) 428–434.
[35] P.J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, J Comput Appl Math 20 (1987) 53–65.
[36] J.L. Bentley, Multidimensional binary search trees used for associative searching, Commun. ACM 18 (9) (1975) 509–517.
[37] J. Wang, X.L. Zhang, Improving pseudo labels with intra-class similarity for unsupervised domain adaptation, Pattern Recognit 138 (2023) 109379, doi:10.1016/j.patcog.2023.109379.
[38] V. Vovk, Universal predictive systems, Pattern Recognit 126 (2022) 108536, doi:10.1016/j.patcog.2022.108536.

Dimitrios Boursinos received his PhD degree in Electrical Engineering from the Vanderbilt University in 2022. He has also been a research assistant in the Institute for Software Integrated Systems. His research interests focus on machine learning and statistical inference.

Xenofon Koutsoukos is currently a Professor and the Chair of the Department of Computer Science and a Senior Research Scientist with the Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA. He is a Fellow of the IEEE. His research focuses on area of resilient cyber-physical systems.