ASSURANCE MONITORING OF CYBER-PHYSICAL SYSTEMS WITH LEARNING ENABLED COMPONENTS

By

Dimitrios Boursinos

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

March 31, 2022

Nashville, Tennessee

Approved:

Xenofon Koutsoukos, Ph.D.

Gabor Karsai, Ph.D.

Janos Sztipanovits, Ph.D.

Abhishek Dubey, Ph.D.

Taylor Johnson, Ph.D.

ACKNOWLEDGMENTS

I would first like to thank my advisor, Professor Xenofon Koutsoukos, for his guidance and support throughout the Ph.D. program as well as Nag Mahadevan and Daniel Stojcsics for their help and great collaboration in my research. I would also like to thank my many other Institute for Software Integrated Systems colleagues and the members of my doctoral committee, Professors Janos Sztipanovits, Gabor Karsai, Taylor Johnson and Ahbishek Dubey.

I would like to acknowledge several of my graduate student peers for their friendship and research collaboration including Ibrahim Ahmed, Dr. Charles Hartsell, Tim Krentz, Shreyas Ramakrishna, Dr. Brad Potteiger, Dr. Feiyang Cai, and Ben Yett among many others.

Finally I would like to thank my family and friends for their continued support and encouragement, including my parents Grigorios and Evangelia Boursinos, and brother Vasilis Boursinos.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

Introduction

I.1 Motivation

Machine learning (ML) components are being used by many cyber-physical system (CPS) applications because of their ability to handle dynamic and uncertain environments. Commonly used ML components are the Deep Neural Networks (DNNs) that are used for perception and decision making tasks in CPS. In autonomous vehicles, for example, perception problems deal with making sense of the surroundings like recognizing correctly traffic signs. Although such components offer many advantages for representing knowledge in high-dimensional spaces and approximating complex functions, they introduce significant challenges when they are integrated into CPSs. Typical DNNs are non-transparent and it is not clear how to rationalize their predictions. Modern architectures are parameterized using million of values which makes reasoning about their predictions very challenging.

The use of DNNs introduce new types of hazards in CPSs that can have disastrous consequences and need to be addressed for engineering trustworthy systems. A DNN is designed using learning techniques that require specification of the task, performance measure for evaluating how well the task is performed, and experience which typically includes training and testing data. Utilizing DNNs for tasks where different degrees of autonomy autonomy is required, presents challenges related to the difficulty of evaluating the risk of the autonomous decisions in tasks that are hard to specify. An example of such task in CPS domain is the perception of the environment. This is a functionality that is difficult to specify, and typically, specifications are based on examples. Modern DNN architectures encode the information learned from the training examples, in a complex manner and it is hard to reason about the encoding and what affects their decisions. Non-transparency is an obstacle to monitoring because it is more difficult to have confidence that the model is operating as intended.

Complementing the predictions of DNNs with a confidence measure can be very useful for improving the trustworthiness of such models and allow their application to safety critical systems. We consider classification applications in CPS. The objective is to complement the prediction of DNNs with a computation of trustworthiness. In this dissertation the trustworthiness measure is approached in different ways with the most common ones being p-values and probabilities. In statistics, the p-value is the probability of obtaining results at least as extreme as the observed results of a statistical hypothesis test, assuming that the null hypothesis is correct. Each of these approaches offer different guarantees and one may be preferred over the

other depending on the application's specifications. We focus on computationally efficient algorithms that can be used for real-time monitoring and decision-making. An efficient and robust approach must ensure a small and well-calibrated error rate while limiting the number of times a trustworthy decisions cannot be made.

I.2 Research Challenges

Cyber-physical systems (CPS) are engineered systems in which physical and software components are deeply intertwined. Most modern products in major industrial sectors, such as automotive, avionics, medical devices, and power systems already are or rapidly becoming CPS driven by new requirements and competitive pressures. These are safety critical fields where the introduction of autonomous components are challenging as potential incorrect decisions can have very serious consequences [2]. For such components to safely be integrated in CPS they need to be complemented by methods and practices that will assure the safe operation of the system.

Semi-autonomous and autonomous vehicles are a very significant domain and opportunity for CPS [3]. Modern vehicles are equipped with a number of sensors, like cameras and Light Detection and Ranging (LiDAR) sensors, to either achieve fully autonomous driving or assist the human operator in dangerous scenarios. Even though all these sensors produce a lot of information about the environment and the surroundings of the vehicle, it can be challenging and tedious to have some desired control actions for all possible scenarios on continuously changing environments. This shows the need for integrating machine learning components in CPS that can deal with decision-making in dynamic environments.

The most commonly used machine learning component is the deep neural networks because of their high knowledge capacity and their ability to receive and make decisions for high-dimensional inputs. They achieve this by transforming the inputs to a number of sequential feature layers of less-and-less abstraction until a classifier can be defined in a lower dimensional feature space. The mapping between the different feature layers is learned using labeled training data and the knowledge of a particular DNN model is stored in its weight parameters. Modern architectures can have hundreds of layers and are parameterized using millions of values. This makes it challenging for DNNs to be used in safety critical CPS application because its hard to understand how a DNN makes a particular decision, or reason about a decision, and how confident a decision is. This problem with DNNs is also called *non transparency*. DNNs commonly use a *softmax* layer to provide probability-like outputs, meaning the output for each class is in [0,1] and the sum of the outputs of all possible classes is 1. However, these probabilities are typically overconfident even for inputs coming from the same distribution as the training data [4] and they cannot be used as reliable confidence measures. This problem is commonly referred as miscalibration, meaning that the confidence scores that compliment decisions are not

accurate indicators of the expected error-rate.

Deploying a CPS with machine learning components in the real world comes with challenges that are hard to foresee in design-time. CPS usually operate in highly dynamic *open worlds* where unknown scenarios may appear. Even using very deep DNN architectures it is hard to have training data for every possible input a system may face during testing. The unpredictability in the real world will force the system to work under different conditions. For example, an autonomous vehicle has to drive safely in different weather conditions. Different operating conditions may introduce different kind of noise into the input data that will make confident decisions harder. There is a detailed work on the influence of rain on LIDAR sensors in [5]. Also different environmental conditions may require different control decisions to achieve safety. Finally, in the real world, input data may provide partial information. In the field of autonomous vehicles inputs usually come as sequences. For example a common task is the *traffic sign recognition*. Traffic signs many times are far for the vehicles to be able to recognize them confidently or covered by obstacles so multiple input frames may be required until a decision can be made. The combination of statistical results computed on subsequent, time-correlated, and high-dimensional inputs presents a lot of challenges for the development of valid and well-calibrated assurance monitors, as well as decision-making.

The last set of challenges has to do with technical issues regarding different kinds of system limitations. Integrating DNNs in CPS comes with restrictions on the computational power. Mobile systems many times have limited memory that will not allow for computationally expensive models to work. Finally, in autonomous CPS it is desirable for human intervention to be as limited as possible. In real-time systems, the decision time is usually very short making it impractical for a human operator to take a decision in time. In such cases, no decision could lead the system into a dangerous state.

We take these areas of research into account when addressing the following research challenges.

- How do we compute well-calibrated confidence metrics with each decision made by a Learning Enabled Component (LEC) for effective integration in CPS?
- How to minimize the human intervention and maximize the safe operation time?
- How confidence metrics computed on time-correlated data can be combined to produce aggregate confidence metrics for a sequence?
- How to process and take confident decisions on high-dimensional data?
- How can the confidence metrics be expressed in a form that is easy to understood and evaluated by humans?

I.3 Research Contributions

For well-calibrated classification methods to be practical in real life problems they need to be able to be used on large and high-dimensional datasets. ICP performs well when used in datasets with a small number of features but does not scale well. Our contributions aim in allowing for predictions with well-calibrated confidence metrics in real life scenarios. The research contributions of this dissertation are presented as a series of publications in Chapters III through VI. The contributions in each of these chapters are:

- In **Chapter III** we present the idea of transforming the high-dimensional inputs into lower-dimensional embedding representations that can be handled efficiently by ICP. For these representations to be used with ICP they need to be in a form in which similarity between different data points can be defined. We use distance metric learning techniques to compute embedding representations such that the euclidean distance between them is a metric of similarity between the original inputs. Moreover, autonomous systems are desirable to operate safely for as long as possible and require as little intervention by humans as possible. We present an optimization method to minimize the instances where a confident decision cannot be made for a given input. Our presented methods have been implemented and evaluated in applications such as traffic sign recognition, speaker recognition and robotic navigation.

- Many applications use input data that are part of sequences, such as still frames belonging to a video footage. Most machine learning techniques, including ICP, operate on the assumption that the data are independent and identically distributed. In **Chapter IV** we propose a method that improve the classification accuracy on sequential data by combining results computed for each individual data point belonging to the sequence. This approach is based on a feedback loop that controls the sensors and queries for a new input until a confident prediction can be made. Our presented methods have been implemented and evaluated in video sequences of traffic sign recognition scenarios as a car approaches traffic signs.

- Decision making in the presence of uncertainty is an important part in the operation of CPS. The first major contribution in **Chapter V** is the use of statistical methods to combine p-values of subsequent inputs and compute aggregate assurance metrics for sequences. Then, the second main contribution is the development of decision making methods that we optimize to maximize the amount of decisions while minimizing the error-rate. Our presented methods have been implemented and evaluated in video sequences of traffic sign recognition scenarios as a car approaches traffic signs.

- The confidence of a Learning Enabled Component on the correctness of a decision is desirable to be in terms that can be understood by humans. In **Chapter VI** the first main contribution is the development

of methods that compute the confidence of decisions in terms of probability intervals that are well-calibrated to represent the actual probability of correctness. The computed probability intervals are evaluated for their calibration, or how well they estimate the true probabilities and efficiency, or how informative they are. We evaluated this method in image recognition applications as well as detection of network attacks. The second main contribution is an extension of probability interval computation that improves the calibration and efficiency of the probability intervals by learning from unlabeled test data during execution.

CPS often are designed for safety-critical applications, with a necessity for producing computation and actuation output in a predictable manner, providing for safe, and consistent operation of the physical process. As such, CPS such as medical devices, automobiles, and military applications are considered real time systems, requiring that underlying computation processes execute within stringent time constraints. We optimize our proposed methods and extensions to minimize their memory and computational requirements to make it possible for them to be used by less powerful computer found in CPS and in real-time.

I.4 Organization

The rest of the proposal will be sequenced as follows. Chapter II introduces the related work needed for better understanding the field of our research, the basis of our approaches as well as other approaches on the same problem in the literature. Chapters III through VI present the research contributions summarized in the previous section. Chapter VII ends with concluding remarks.

CHAPTER II

Related Work

CPS applications use learning enabled components (LECs) for various tasks, like perception and control. Each decision taken by an LEC must be associated with an assurance metric that quantify the uncertainty of such decision. This metric must be well-calibrated, meaning it should be an accurate representation of the actual probability that a decision is correct or it must bound the expected error-rate. Modern machine learning components tend to not be well-calibrated. However this property is essential for many systems and therefor many researchers have worked in this area trying to develop methods that compute well-calibrated assurance metrics for existing ML components. Related research in well-calibrated ML components follows in Section II.1.

Most of our applications are related to self-driving vehicles and other mobile robots, applications that add some restrictions on the available computational power. Self-driving vehicles are typically equipped with cameras generating a large amount of data that have to be processed in real-time. Many of the approaches that calculate accurate assurance metrics do not perform well with high-dimensional inputs or they are not fast enough for real-time applications. For this reason it is essential to compute appropriate low-dimensional representations for the high-dimensional original inputs. Related work in computing appropriate representations is in Section II.2. Moreover, when the inputs to the system is produced by cameras, usually there are particular objects on the frame that affect the decision-making process. Object detection has gotten a lot of attention lately and we present related work in Section II.3.

The related work starts with describing what assurance means with regards to machine learning components and why it is needed. There is a review of a number of methods that have approached this problem and there is a more lengthy description of the Conformal Prediction framework which is used extensively in our work. Next there is an overview on distance metric learning methods that are used when a metric of similarity is needed between high-dimensional data pairs. Afterwards, object detection methods are described. It is only the last ten years that deep neural networks are used for this problem but for better understanding of the problem and the proposed approaches the literature review starts with earlier attempts and continues to the modern deep learning architectures used in our work.

II.1 Assurance in Machine Learning

Modern deep neural network architectures tend to be poorly calibrated [4, 6, 7]. Neural networks for classification typically use a softmax layer to produce a probability-like output for each class. The chosen class

is the one with the highest probability, however this generated probability measure is often higher than the actual posterior probability that the prediction is correct. Other factors that affect the calibration in DNNs are the depth, width, weight decay and Batch Normalization [4].

CPS that use machine learning components for perception and control can benefit from well-calibrated classifiers. Accurate error-rate bounds provide assurance guarantees in safety-critical applications but also make the decision confidence interpretable by humans. The significance of accurate confidence metrics was recognized very early and the earliest work we know of is in the context of forecasting [8, 9]. Modern classifiers that produce softmax outputs are generally overconfident and several methods have been proposed that compute scaling factors for calibration. These methods are presented in subsection II.1.2. First it is important to see how calibration can be evaluated.

II.1.1 Notations and Evaluation Metrics

Consider a dataset $\{z_1, \ldots, z_l\}$ of examples, where each $z_i \in Z$ is a pair $(x_i, y_i) \in X \times Y$ with x_i the feature vector and y_i the label of that example. We also consider the classifier $f : X \to \mathbb{R}^{|Y|}$ with output $o = f(x)$ scores. A decision is made as $\hat{y} = \text{argmax}_i(o_i)$ and has confidence $\hat{p} = \max_i(o_i)$. A classifier is considered *calibrated* when the confidence $\hat{p}$ represents the true probability that the prediction $\hat{y}$ is correct. This means that given 100 predictions, each of them having a confidence of 0.95, we expect 95 of them to be correctly classified. Mathematically, calibration is defined as:

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

The calibration as it defined by Eq. II.1 can only be approximated empirically using finitely many samples as $\hat{p}$ is a continuous random variable.

A common way to visually inspect the calibration of different models is the *reliability diagrams* [9, 10]. These are histograms that plot the expected sample accuracy as a function of confidence. A perfectly diagonal line, where all values of accuracy are equal to their associated confidence values is an indication of a perfectly calibrated model while deviations in the plotted line are signs of miscalibration. In order to plot the expected accuracy as a function of the confidence using a finite amount of samples we split the samples into M interval bins of size $\frac{1}{M}$ based on their confidence values. Then the expected accuracy for a set of samples B_m in bin $I_m = \left(\frac{m-1}{M}, \frac{m}{M}\right)$ is:

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} (\hat{y}_i = y_i)$$

The average confidence within B_m is computed as:

$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i$$

$\text{acc}(B_m)$ and $\text{conf}(B_m)$ are approximations of the left and right hand side of Eq. II.1 and a well calibrated model will have $\text{acc}(B_m) = \text{conf}(B_m)$ for all bins $m \in \{1, \ldots, m\}$.

The reliability diagrams make it easy to qualitatively get an idea of the model's calibration and observe whether there are samples with specific confidence values that show larger miscalibration. However, when comparing the calibration between different models it is useful to have a single scalar as an evaluation metric for calibration. One way to summarize the calibration as it is defined Eq. II.1 for a large number of samples using a single scalar is to compute the expected value $\mathop{\mathbb{E}}_{\hat{p}}[|\mathbb{P}(\hat{y} = y|\hat{p} = p) - p|]$. The empirical computation of this expected value is called *Expected Calibration Error* (ECE) [11] and is a similar idea like the reliability diagrams. The samples are split into M equally-spaced interval bins of size $\frac{1}{M}$ based on their confidence values. Then ECE is the following weighted average:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)|, \tag{II.2}$$

where n is the total number of samples. The lower the value of ECE, the better the calibration of the model.

ECE is frequently used as an evaluation metric by many calibration methods in the literature because of its simplicity to summarize Eq. II.1. On the other hand, many times in safety critical applications it more useful to compute the maximum miscalibration of a model than the mean value. This metric is called Maximum Calibration Error (MCE) [11] and is computed as:

$$\text{MCE} = \max_{m \in \{1,\ldots,M\}} |\text{acc}(B_m) - \text{conf}(B_m)| \tag{II.3}$$

II.1.2 Calibration Methods

Having defined metrics for evaluating different calibration methods we can review existing calibration methods. The available calibration methods generally belong in two categories: parametric and non-parametric. The parametric methods assume that the probabilities follow certain well-known distributions whose parameters are to be estimated from the training data. The Platt's scaling method [12] is proposed for the calibration of Support Vector Machine (SVM) outputs. After the training of an SVM, the method computes the parameters of a sigmoid function to map the non-probabilistic outputs into probabilities. The Platt's scaling method has also been used for neural network models in a similar manner [10]. Piecewise logistic regression is an

extension of Platt scaling and assumes that the log-odds of calibrated probabilities follow a piecewise linear function [13]. Platt's scaling method was originally purposed for binary classifiers. A variant of this that can be used with multiclass classifiers is the temperature scaling [4] which can be applied in DNNs with a softmax output layer. After the training of the DNN, a temperature scaling factor T is computed on a validation set to scale the softmax outputs. However, while temperature scaling achieves good calibration when the data in the validation dataset are independent and identically distributed (IID), there is no calibration guarantee under distribution shifts [14]. Experiments in [15] show that Platt's scaling and temperature scaling are not as well calibrated as it is reported and it is difficult to know how miscalibrated they are.

Histogram binning or quantile binning is a commonly used non-parametric approach with either equal-width or equal-frequency bins. It divides the outputs of a classifier into bins and computes the calibrated probability as the ratio of correct classifications in each bin [16]. Isotonic Regression is a generalization of histogram binning by jointly optimizing the bin boundaries and bin predictions [17]. An extention of isotonic regression is a method called ensemble of near-isotonic regression (ENIR) that uses selective Bayesian averaging to ensemble the nearly-isotonic regression models [18]. Adaptive calibration of predictions (ACP) also uses the ratio of correct classifications as the posterior probability in each bin, but it obtains bins from a 95% confidence interval around each individual prediction [19]. Estimating calibrated probabilities is a more significant issue in class imbalance and class overlap problems. Receiver Operating Characteristics (ROC) Binning uses the ROC curves to construct equal-width bins that provide accurate calibrated probabilities that are robust to changes in the prevalence of the positive class [20]. Bayesian binning into quantiles (BBQ) extends the simple histogram-binning calibration method by considering multiple equal frequency Histogram Binning models and their combination as the calibration result [11].

Another framework developed to produce well-calibrated confidence values is the Conformal Prediction (CP) [21–23]. The conformal prediction framework can be applied to produce calibrated confidence values with a variety of machine learning algorithms with slight modifications. Using CP together with machine learning models such as DNNs is computationally inefficient. In [24], the authors suggest a modified version of the CP framework, Inductive Conformal Prediction (ICP) that has less computational overhead and they evaluate the results using DNNs as undelying model. Deep k-Nearest Neighbors (DkNN) is an approach based on ICP for classification problems that uses the activations from all the hidden layers of a neural network as features [25]. The method is based on the assumption that when a DNN makes a wrong prediction, there is a specific hidden layer that generated intermediate results that lead to the wrong prediction. Taking into account all the hidden layers can lead to better interpretability of the predictions. In [26], the authors present an empirical investigation of decision trees as conformal predictors and analyzed the effects of different split criteria, such as the Gini index and the entropy, on ICP. There are similar evaluations using ICP

with random forests [27, 28] as well as SVMs [29]. The above methods are applied to datasets and show good results when the input data are IID. In [30] we showed that ICP under-performs when the input data are sequential. Individual frames of a sequence might contain partial information regarding the input and more frames might be needed for ICP to reach a confident prediction. The performance of ICP in this case can be improved by designing a feedback-loop configuration that queries the sensors until a single confident decision can be reached.

Confidence bounds can also be generated for regression problems. In this case instead of sets of multiple candidate labels we have intervals around a point prediction that include the correct prediction with a desired confidence. There are ICP methods for regression problems with different underlying machine learning algorithms. In [31], the authors use the k-Nearest Neighbours Regression (k-NNR) as a predictor and evaluate the effects of different nonconformity functions. Random forests can also be used in regression problems. In [32], there is a comparison on the generated confidence bounds using k-NNR and DNNs [33]. An alternative framework used to compute confidence bounds on regression problems is the Simultaneous Confidence Bands. The method presented in [34] generates linear confidence bounds centered around the point prediction of a regression model. In this approach, the model used for predictions has to be estimated by a sum of linear models. Models that satisfy this condition are the least squares polynomial models, kernel methods and smoothing splines. Functional Principal Components (FPC) analysis can be used for the decomposition of an arbitrary regression model to a combination of linear models [35].

II.1.3 The Conformal Prediction Framework

The conformal prediction (CP) framework [21, 23] uses past experience in the form of collected data to determine the confidence in the prediction of a new unseen input. Unlike the above methods that attempt to compute well-calibrated confidences that approximate the real probabilities of an accurate prediction, in CP the desired confidence is a parameter that can be chosen. Once the confidence, or the *significance level* as it is usually called, is set, the CP framework produces a set a set of labels that includes the correct class with the chosen confidence.

More formally consider a training set $\{z_1, \ldots, z_l\}$ of examples, where each $z_i \in Z$ is a pair (x_i, y_i) with x_i the feature vector and y_i the label of that example. The parameterization of assurance is the significance level $\varepsilon \in [0, 1]$ and it is chosen to bound the error-rate for each individual prediction. The *confidence level* can be defined as $1 - \varepsilon$. Γ^ε is a *set predictor* parametrized by ε so that for an unseen test input x_{l+1} the probability it will not include the correct class y_{l+1} is less than ε. The set predictor is valid if:

$$\mathbb{P}(y_{l+1} \notin \Gamma^\varepsilon) \leq \varepsilon \tag{II.4}$$

where y_{l+1} is the correct class for input x_{l+1}. It can be seen that the validity property can be achieved with the trivial set predictor that include every possible class $\Gamma^{\varepsilon} = \boldsymbol{Y}$. However this does not offer any information toward the decision-making process. *Efficiency* of a set predictor is a measure of the number of classes it includes. Neither validity nor efficiency are enough by themselves as the one does not guarantee the other. We are looking for the most efficient prediction set among all the valid ones which in turn provides the most information toward a final prediction with a chosen confidence. A confidence predictor is nested in the sense that for $0 \leq \varepsilon_1 \leq \varepsilon_2 \leq 1$, $\Gamma^{\varepsilon_1} \supseteq \Gamma^{\varepsilon_2}$ [23]. For smaller significance level values, set predictors are expected to include more classes.

According to [23] CP makes two kinds of assumptions about the way the examples z_i, $i = 1,\ldots,l+1$ are generated. Under the *randomness assumption*, the $l+1$ examples are generated independently from the same unknown probability distribution. Under the *exchangeability assumption* the sequence z_i, $i = 1,\ldots,l+1$ is generated from a probability distribution that is exchangeable: for any permutation π of the set $\{1,\ldots,l+1\}$, the distribution of the permuted sequence $(z_{\pi(1)}, \ldots, z_{\pi(l+1)})$ is the same as the distribution of the original sequence $(z_1,\ldots,z_{l+1})$.

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. For a given test example z_{l+1} with candidate label $\tilde{y}_{l+1}$, a nonconformity function $\alpha(x_{l+1}, y_{l+1})$ assigns a numerical score indicating how different the example z_{l+1} is from the examples in $\{z_1,\ldots,z_l\}$. Nonconformity functions can be defined in different ways [21–23,25,36–38]. For example, a nonconformity function can be defined as the number of the k-nearest neighbors to z_{l+1} in the training set that are labeled different than the candidate label $\tilde{y}_{l+1}$ (k-nearest neighbors nonconformity measure). The labels of the k-NN to x_{l+1} are stored in a multi-set Ω. The k-NN nonconformity of input x with a candidate label y is defined as

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

In a similar manner, instead of counting the number of labels of the k-NN that are the same as the candidate label y of a sample x, we can count how many of the k-NN of label y are are within a certain range. We can define the *Number of Close Examples* (NCE) [38] as 1 minus the ratio of the k-NN of x that belong to the same class as the candidate class that are within a certain proximity range

$$\alpha(x,y) = 1 - \frac{|d(x,x_i) \leq \theta : y_i = y|}{k}$$

where $x_i : i = 1,\ldots,k$ the k-NN of x that are labeled the same as x and d is a distance metric between samples

x.

The 1-NN NCM requires to find the most similar example of a test input x in the training set that is labeled the same as the candidate label y as well as the most similar example in the training set that belongs to any class other than y and is defined as

$$\alpha(x,y) = \frac{\min_{i=1,\ldots,n:y_i=y} d(x,x_i)}{\min_{i=1,\ldots,n:y_i\neq y} d(x,x_i)}$$

where d is a distance metric between input examples.

A NCM can be defined in a similar way but instead of using only the closest neighbor to a sample, we can use k-NN. The Relative Neighborhood Distance (RND) [38] can be defined as

$$\alpha(x,y) = \frac{\sum_{i=1}^{k} d(x,x_i)}{\sum_{j=1}^{k} d(x,x_j)}$$

where $y_i = y : i = 1,\ldots,k$, $y_j \neq y : j = 1,\ldots,k$ and d is a distance metric between input examples.

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

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

It should be noted that for computing the nearest centroid NCM, CP needs to store only the centroid for each class.

A nonconformity function does not need to be defined in the input space and take as input the examples in their original form. Features can be extracted from the input samples and then be used as inputs for the NCM computation. One such way of defining an NCM is the *Deep Nearest Neighbors* (DkNN) NCM [25]. This way of computing the NCM is motivated by the architectural design of neural networks in layers. Each layer deeper in the hierarchy is an increasingly abstract representation of the input domain [39]. These representations are also called *embeddings*. The last layer is sufficiently abstract for a linear decision function to be used for classification. The idea is similar to the k-NN NCM function, only here we first compute the embedding representations of the input on every layer of a DNN and then compute the k-NN for every layer representation. For such k-NN to be computed for every layer, all the embedding representations for the training data $z_1,\ldots,z_l$ need to be computed and stored using the same trained DNN classifier that is used to compute the embedding representations of the test sample z_{l+1}. The nonconformity of an input x with the

label y is defined as:

$$\alpha(x,y) = \sum_{\lambda \in 1...n} |i \in \Omega_n : i \neq y|.$$

where n is the total number of layers and Ω_n the multi-set of labels for the training samples whose representations are closest to the test input's at layer λ. Because the output of the layers is often high-dimensional the authors used Locality-Sensitive Hashing (LSH) [40–42] to find the nearest neighbors according to the cosine similarity between vectors. Unlike the most common uses of hash functions, LSH is designed to maximize the collision between similar samples. The use of all the layers has better robustness when computing the NCM when input data are out of distribution even though more memory is required to store the representations of the training data.

Another family of NCMs is the *model agnostic* nonconformity functions [36]. Unlike the previous NCMs that are defined either on the input domain or the embedding representations domain, the model agnostic NCMs are based on probability estimates which can be computed by different kind of underlying models like decision trees [26], random forests [43] or the output layer of a DNN, the softmax layer that produces probability like estimates for each class. The *Hinge* [43] is based simply on the probability estimate provided for the correct class label y assigned to sample x:

$$\alpha(x,y) = 1 - \hat{P}(y|x)$$

where $\hat{P}$ is the estimated probability output.

Margin [26] is an NCM that considers two class labels: the true class label and the most likely incorrect class label. The margin NCM is defined as:

$$\alpha(x,y) = \max_{y_j \neq y} \hat{P}(y_j|x) - \hat{P}(y|x)$$

meaning a nonconforming example is one which has a low probability estimate for the true class label and/or a high probability estimate for any other (incorrect) class label.

The last model-agnostic NCM is the *Brier score* [44] which considers all possible class labels:

$$\alpha(x,y) = \frac{1}{|Y|} \sum_{y_j}^{Y} \left(P[y_j|x] - \hat{P}[y_j|x] \right)^2$$

where $P(y|x) = 1$ if $y_j = y$ and 0 otherwise. Here the nonconformity of an example depends on the probability estimates of all classes and even small differences regarding the correct class will affect the final score.

Another proposed way of computing the NCMs is using Support Vector Machines (SVMs). This underly-

ing algorithm has been used in different applications like medical [45] and chemoinformatics [46]. SVMs are trained to classify data by computing some optimal hyperplanes to separate the different classes. It is natural to compute the nonconformity score of an example as a function of its distance to the separating hyperplane. However, the use of SVMs can introduce a number of challenges depending on the application, like:

1. The dataset may be to large to be handled by an SVM
2. Imbalanced classes
3. The choice of the appropriate kernel

Training SVMs on large datasets is generally challenging, as storing the kernel matrix requires memory that scales quadratically with the number of data points [47]. Similar increase is observed in the training time as well [47]. This problem has received a significant amount of research and there has been proposed different methods for SVM training [47–54]. SVMs work effectively on balanced datasets but are sensitive to imbalances and produce sub-optimal models in such situations. There are different reasons for this. When training on imbalanced data the separating hyperplane can be skewed towards the minority class and can cause the generation of more false negative predictions [55, 56]. Another source of boundary skew is that the ratio between the positive and negative support vectors becomes more imbalanced with the increase in the imbalance of the dataset [57,58]. As a result a test sample close to the boundary is more likely to be dominated by negative support vectors and be classified as negative. Finally the choice of an appropriate kernel depends on the particular application and the used features. In [46] the authors used the Tanimoto Similarity [59] as a kernel and compared it with a kernel consisting of the composition the Tanimoto similarity with Gaussian RBF.

Other underlying models that have been used for the computation of nonconformity scores are the *Multinomial Naive Bayes* [46], *decision trees* [26] and *random forests* [38, 43]. The Multinomial Naive Bayes classifier is suitable for classification when the features are discrete. Its advantages are the simplicity and fast training. A nonconformity function can be defined as $\alpha(x,y) = -\log p(y|x)$, where p is the posterior probability estimated by Naive Bayes. When the standard decision trees produce probabilities instead of just the class labels of the decision, they are refered as Probability Estimation Trees (PETs) [60]. PETs' output probabilities, just like SVMs' and DNNs', are not well-calibrated [16]. In [26] the authors propose a way to define an NC function based on decision-trees using the *Margin* NC function above. They show that the decision trees trained to be used as part of the nonconformity framework should use no pruning have smoothed probability estimates. The choice of split-criterion seemed to not have any significant effect in the performance of CP. Random forests [61] consist of a large number of individual decision trees that operate

as an ensemble. RFs are computationally efficient since each tree is built independently of the others. Each tree makes a class prediction and the class with the most votes becomes the model's prediction. With a large number of decision trees this model is robust to overfitting and noise in the data [61]. The simplest way to compute a NC score using RFs for an exampe x is 1 minus the ratio of trees that vote for the actual class label y [38, 43]:

$$\alpha(x,y) = 1 - \frac{|y_i = y : i = 1,\ldots,n|}{n}$$

where n is the number of decision trees and y_i the decision of the tree i. Finally, RFs can be used to compute *proximities* between pairs of examples. This is a measure of similarity between input samples and can be used to compute the NCMs that require the computation of distance between examples like the NCE, 1-NN, RND and Nearest Centroid.

The nonconformity score is an indication of how uncommon a test input is compared to the training data. Input data that come from the same distribution as the training data will produce low nonconformity scores and are expected to lead to more confident classifications while unusual inputs will have higher nonconformity score, indication of less confident classifications. However, this metric by itself does not tell us how unusual a new example is with respect to the training set. In order to convert the nonconformity scores into well-calibrated probabilities we use the notion of *p-values*. The first way to compute the p-value assigned to a sample x_{l+g} with a candidate label y_{l+g} is called *Transductive Conformal Prediction* (TCP) [23,62]. Consider all possible classifications $Y_1,\ldots,Y_c$. For all the training data as well as the new test input compute the nonconformity score with regards to all possible classifications:

$$\begin{gathered}
\alpha_1^{(Y_1)},\ldots,\alpha_l^{(Y_1)},\alpha_{l+g}^{(Y_1)} \\
\vdots \\
\alpha_1^{(Y_c)},\ldots,\alpha_l^{(Y_c)},\alpha_{l+g}^{(Y_c)}
\end{gathered}$$

The p-value assigned to x_{l+g} been classified as Y_j is:

$$p(x_{l+g},Y_j) = \frac{\#\{i = 1,\ldots,l,l+g : \alpha_i^{(Y_j)} \geq \alpha_{l+g}^{(Y_j)}\}}{l+1} \tag{II.5}$$

This method of computing the p-values is relatively inefficient as it involves the computation of large number of nonconformity scores. Depending on the choice of underlying model this may not be feasible.

The second way of computing the p-values is called *Inductive Conformal Prediction* (ICP) [23, 62]. In this method, the training set is split into the *proper training set* $(z_1,\ldots,z_m)$ of size $m < l$ and the *calibration*

set $(z_{m+1},\ldots,z_l)$ of size $l-m$. This method of computing the p-values for a new test input x_{l+g} is more efficient as it requires only the computation of the nonconformity scores of the data in the calibration set with respect to their ground truth labels, $\alpha(x_i,y_i), i=m+1,\ldots,l$. The same underlying model is used to compute $\alpha(x_{l+g},Y_j), j=1,\ldots,c$, the nonconformity scores of the test input with respect to all possible labels. Then the p-value for the pair (x_{l+g},Y_j) is:

$$p(x_{l+g},Y_j)=\frac{\#\{i=m+1,\ldots,l : a_i \geq \alpha_{l+g}^{(Y_j)}\}+1}{l-m+1} \tag{II.6}$$

The computed p-values, either by Eq. (II.5) or by Eq. (II.6) are then used to form the set predictors Γ^{ε} that satisfy the validity property of Eq. (II.4). After the desired significance level ε has been chosen a candidate label Y_j is added to Γ^{ε} if $p(x_{l+g},Y_j)>\varepsilon$. It is shown in [23] that the prediction sets computed by ICP are valid, that is, the probability of error will not exceed ε for any $\varepsilon \in (0,1)$ for any choice of the nonconformity function.

II.2 Distance Metric Learning

Many real-world applications use different distance metrics in various tasks. For example in computer vision such tasks are image classification and content-based image retrieval (CBIR). Such applications require a way to quantify the similarity between different images. Then a k-nearest-neighbors (k-nn) classifier can be used to identify similar images. The metric chosen to define the distance between images can strongly affect performance. With that, there is a technical problem that should be considered that has to do with the computational requirements for these tasks. Using k-nn on a high-dimensional space that represents for example images, is inefficient and require a large memory capacity to store all available images 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 that can be used easier on similarity-based data retrieval tasks. The task of dimensionality reduction is to find a small number of features to represent a large number of observed dimensions.

There are many proposed approaches on learning appropriate distance metrics for similarity estimation.We will overview the ones that belong in *supervised distance metric learning*, which is related to our proposed methods. The supervised distance metric learning require the collected training data to be labeled. However, unlike training a classifier where each data point is assigned to a ground truth label and we minimize a loss function so that the classification will be the same as the ground truth label, in distance metric learning, the data points are considered in pairs. The associated loss function is defined using pairwise constraints such that its minimization will make data points belonging to the same classes be close to each other

and data points belonging to different classes be far from each other. The constraints can be defined either globally, on all possible pairwise distances, or locally on a sub-region of the embedding space.

II.2.1 Early Work

Many distance metric learning approaches have been proposed for applications where the available data are labeled and the objective is to keep all the data points of the same classes closes to each other and far apart from data belonging to different classes. Suppose we have a collection of data points

$$C = \{x_i\}_{i=1}^{m} \subseteq \mathbb{R}^n.$$

Unlike training of classifiers where the label of each data point is a particular class, in distance metric learning pairs of data points are annotated either as semantically-similar or as semantically-dissimilar. The learned metric should place similar data points close to each other and far from the dissimilar ones. Suppose we are given the similarities information as:

$$S: \quad (x_i, x_j) \in S \quad \text{if } x_i \text{ and } x_j \text{ are similar}$$

and

$$D: \quad (x_i, x_j) \in D \quad \text{if } x_i \text{ and } x_j \text{ are dissimilar}$$

The distance metric between two data points x and y can be written as:

$$d_M(x,y) = \|x-y\|_M = \sqrt{(x-y)^T M (x-y)}, \qquad \text{where } M \in \mathbb{R}^{n\times n}. \tag{II.7}$$

For the distance metric to satisfy the the non-negativity and triangle inequality properties, M needs to be positive, semi-definite, $M \succeq 0$. M parameterizes a family of Mahalanobis distances over $\mathbb{R}^n$ and setting $M = I$ gives the Euclidean distance. The convex optimization problem for distance metric learning subject to the constraints in S and D is defined in [63] as a semi-definite programming problem [64]:

$$\begin{aligned} \min_{M} \quad & \sum_{(x_i,x_j)\in S} \left\|x_i - x_j\right\|_M^2 \\ \text{s.t.} \quad & \sum_{(x_i,x_j)\in D} \left\|x_i - x_j\right\|_M \geq 1, \\ & M \succeq 0. \end{aligned}$$

However, this problem cannot be solved efficiently as it does not fall in any special class of semi-definite programming problems. Numerical analysis, such as the Newton-Raphson method can be used to compute the distance metric M but this is a very computational intensive task. In [63] the authors suggest the use of gradient descent and iterative projections [65] to transform the problem to minimizing a quadratic objective subject to a single linear constraint which can be solved efficiently. This solution is still challenging to be applied in high-dimensional problems.

In [66] the proposed method parameterizes $M = A^T WA$, where A is any fixed and known real matrix, and W is a diagonal matrix with non-negative entries. Eq. II.7 then becomes:

$$d_M(x,y) = \|x-y\|_M = \sqrt{((x-y)^T A)W(A^T(x-y))}. \tag{II.8}$$

We see that the role of A is to apply a linear transformation to the input data so that the distance metric becomes a Euclidean distance between the transformed inputs. Using this transformation, the optimization problem can be formalized based on triplet constraints as a convex problem that can be solved efficiently:

$$\min_W \ \|M\|_F^2 + C\sum_{i,j,k}\xi_{ijk}$$

$$\text{s.t.} \quad d_M^2(x_i - x_k) - d_M^2(x_i - x_j) \geq 1 - \xi_{ijk} \ \ \forall (x_i, x_j, x_k) \in \mathbb{R}$$

where $\|M\|_F^2$ the squared Frobenius norm of M, ξ_{ijk} are slack variables like in Support Vector Machine (SVM) classifiers, and $C \geq 0$ is a regularization parameter [67]. The main drawbacks of this approach is that A must be chosen manually, and the distance metric W is quit simple as it only learns a weighting of the features.

In addition to the above general purpose algorithms, many approaches are designed to learn appropriate distance metrics for the k-nn classifier in a local way. This means that the constraints are defined in such a way so that the k-nn of any training data point should belong to the same class and be far from training data that belong to other classes. One of the most well-known such methods for Mahalanobis distance metric learning is the Large Margin Nearest Neighbors (LMNN) [68] and has been the base for many other distance learning methods [69–72]. The distance is learned solving the following optimization problem which is written as a semidefinite (SDP) program:

$$\min_M \quad \sum_{(x_i,x_j)\in S} d_M^2(x_i,x_j) + \mu\sum_{i,j,k}\xi_{ijk}$$

$$\text{s.t.} \qquad d_M^2(x_i,x_k) - d_M^2(x_i,x_j) \geq 1 - \xi_{ijk}$$

where $(x_i, x_k) \in D$, and $\mu \in [0,1]$ is a parameter that controls the push \pull between the training data points, typically set through cross-validation. xi_{ijk} are slack variables like in SVM classifiers. The authors propose a solver to optimize the matrix M. Other solvers have been proposed in [73–76].

The proposed methods presented above compute a linear transformation of the original data by computing a distance metric matrix M. This is because such linear methods can be optimized easier when formalized as SDPs and they are less likely to overfit. However, when the data have a nonlinear structure it is impossible to compute linear distance metrics. Two very common general approaches in learning nonlinear patterns are the SVMs and the DNNs. The idea behind the use of SVMs is to learn a nonlinear transformation of the original data to a feature space where linear distance metrics can be applied. In [77] the authors propose a nonlinear mapping of the original data to either a high-dimensional or a low-dimensional feature space using a kernel function and they use the Mahalanobis distance in this space. The objective is to collapse all examples of the same class to a single point and push examples in other classes infinitely far away. Similarily to the previous method, the Pseudometric Online Learning Algorithm (POLA) [78] attempts to learn a metric that shrinks distances between similarly labeled inputs and expands distances between differently labeled inputs. However unlike [77] differently labeled inputs will be encouraged to be a specific distance away from each other given by a margin. POLA can be implemented given a training set, but it can also be used online when inputs arrive in pairs one after the other. POLA attempts to learn a Mahalanobis metric M and a scalar threshold b such that similarly labeled inputs are at most a distance of $b-1$ apart, while differently labeled inputs are at least a distance of $b+1$ apart.

II.2.2 Deep Learning Methods

In recent years deep neural networks (DNNs) have been used extensively in machine learning tasks because of their ability to compute layers of representations of the input data which can then be used to distinguish between available classes [39, 79]. Features of the input data extracted by DNNs can then be used in different tasks as lower-dimensional representations of the original inputs [80–84] including in distance metric learning. Like with SVMs, DNNs compute nonlinear transformations but they can handle high-dimensional input data easier. The benefits of DNNs over SVMs for nonlinear distance metric learning can be seen in the results of [85]. The authors present a way for unsupervised pre-training of an encoder DNN. After the pre-training, the parameters of the last layer are fine-tuned using a Neighbourhood Component Analysis (NCA) objective function. There have been proposed different ways to train DNNs to produce appropriate representations for distance metric learning. The *Siamese Networks* are formed using two copies of the same DNN architecture that share the same weights as shown in Figure II.1a. They are trained to produce embedding representations that will minimize a distance metric between input pairs that belong to the set S and maximize the distance

between pairs of the set D. In [86] a siamese network was trained for signature verification. The chosen Convolutional Neural Network (CNN) architecture was trained to produce embeddings with small angle between them (cosine=1.0) for pairs of genuine signatures and a large angle (cosine=-1.0) if one of the signatures was a forgery. The Mean Squared Error (MSE) loss function was used for training as described in [87]. In [88] the distance metric used for the semantic similarity estimations was the L_1 distance. The more significant difference between this and the method used in [86] is the loss function minimized by the training process. The loss function is derived from the discriminative learning framework for energy-based models (EBM) and has the form of a contrastive loss function. It is computed to be

$$L(W,y,x_1,x_2) = (1-y)\frac{2}{Q}\|D_w\|^2 + (y)2Qe^{-\frac{2.77}{Q}\|D_w\|}$$

where $D_w = r_1 - r2$, r_1, r_2 the embedding representations of inputs x_1, x_2 and the constant Q is set to the upper bound of $\|D_w\|$. A similar siamese network was used in [89]. In this approach the embedding representations were chosen to be in a space where the euclidean distance is used for similarity estimation. The loss function is, again, in the form of the contrastive loss function

$$L(W,y,x_1,x_2) = (1-y)\frac{1}{2}\|D_w\|_2 + (y)\frac{1}{2}max(0, m - \|D_w\|_2)$$

where y is a binary flag equal to 0 if the inputs x_1 and x_2 are semantically similar and equal to 1 otherwise. m is a margin parameter. In particular, when x_1 and x_2 are not similar, $L = 0$ when $\|D_w\|_2 \geq m$, otherwise the parameters of the network are updated to produce more distant representation for those two elements. The reason behind the use of the margin is that when the distance between pairs of different classes are large enough and at most m, there is no reason to update the network to put the representations even further away from each other and instead focus the training on harder examples.

The *Triplet Network* [90] is a different DNN architecture for distance metric learning. The main difference between this approach and the previous approaches is that for the training it requires triplets of examples, instead of pairs, that are the inputs to three copies of the same DNN with shared weights. More specifically, the triplets are formed using an anchor example x, a positive example x^+ that is semantically similar to x and a negative example x that is semantically different to x. The general architecture can be seen in Figure II.1b. Similarly to the siamese network, the triplet network is used to produce embedding representations on a space where the computation of distance is a measure of similarity. The distance metric that is mostly used in the literature is the Euclidean distance. Different loss functions have been proposed for training of triplet

networks. The FaceNet [91] trains the triplet networks by minimizing the following loss function:

$$L = \sum_{i}^{N} \max\left(\left\| r - r^{+} \right\|_2^2 - \left\| r - r^{-} \right\|_2^2 + m, 0 \right) \tag{II.9}$$

where m is a margin that is enforced between positive and negative pairs as we want $\|r - r^{+}\|_2^2 + m < \|r - r^{-}\|_2^2$. Generating all possible triplets is computationally expensive and results in many triplets that satisfy the goal above. These triplets would not produce gradients and would not contribute to the training and result in slower convergence, as they would still be passed through the network. The training can be accelerated by choosing *hard* triplets that satisfy $\|r - r^{-}\|_2^2 < \|r - r^{+}\|_2^2 + m$ that will result in large gradients. Selecting the hardest negatives can in practice lead to bad local minima early on in training. To avoid this the authors suggest mining for *semi-hard* triplets that satisfy $\|r - r^{-}\|_2^2 < \|r - r^{+}\|_2^2$. The same triplet architecture is used in [90]. In this approach the training of the triplet network is expressed as a 2-class classification problem using softmax layer. More specifically the 2-classes take values:

$$d_{+} = \frac{e^{\left\| r - r^{+} \right\|_2}}{e^{\|r - r^{+}\|_2} + e^{\|r - r^{-}\|_2}} \quad \text{and} \quad d_{-} = \frac{e^{\left\| r - r^{-} \right\|_2}}{e^{\|r - r^{+}\|_2} + e^{\|r - r^{-}\|_2}}$$

Then the MSE is used as a loss function comparing the (d^{+}, d^{-}) with the vctor $(1,0)$ for similar pairs and $(0,1)$ for different ones. Motivated by the triplet loss function (Eq. II.9) and the center loss [92], the authors in [93] propose the *Triplet Center Loss* TCL. In the previous triplet architectures, the similarity and dissimilarity information was embedded into the triplet sample generation. The TCL on the other hand uses single labeled examples as well as the centroids of each class and tries to efficiently minimize the intra-class distances of the learned features as well as maximize the inter-class distances of the deep features simultaneously. The loss function is defined as:

$$L_{\text{TCL}} = \sum_{i=1}^{M} \max\left(\left\| r_i - c_{y_i} \right\|_2 - \min_{j \neq y_i} \left\| r_i - c_j \right\|_2 + m, 0 \right)$$

where $c_1, c_2, \ldots, c_{|Y|}$ the centers of all classes and m the margin parameter similar to the triplet center loss. The TCL can also be combined with the softmax loss, which is commonly used for classification training, defining it as $L_{\text{total}} = \lambda L_{\text{TCL}} + L_{\text{softmax}}$, where λ is a hyper-parameter which controls the trade-off between the TCL and the softmax loss.

II.3 Object Detection

In many CPS applications the control actions are taken based on some visual inputs. For example an autonomous vehicle decides about control commands like steering, throttle and brake by observing its surround-

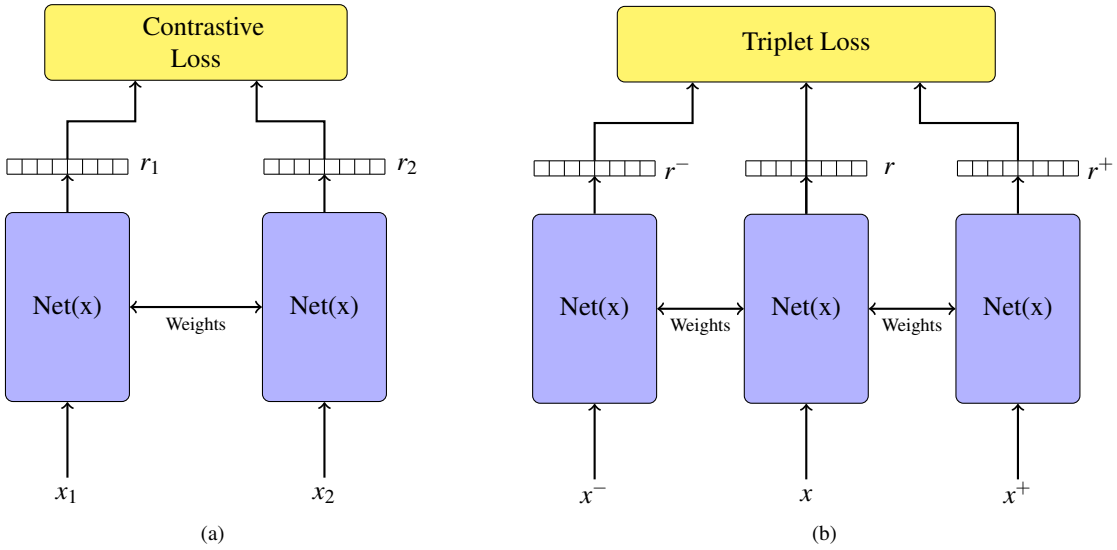


Figure II.1: (a) Siamese network architecture and (b) Triplet network architecture

ings with cameras. The presence of different objects of interest, like other vehicles, traffic signs, pedestrians, and their location in the frame can lead to different control actions. Other CPS applications that require localization of objects in images include and are not limited to robot vision, security, human computer interaction, intelligent video surveillance and object tracking. *Object detection* is a fundamental problem in computer vision and the objective is to locate and identify objects that belong to some given categories in images. Because of the importance of object detection in applications that use visual inputs, this problem has received a lot of attention. The first attempts [94] were based on geometric representations to identify objects. The recent advances in deep learning [39] has lead to the development of multilayer DNN architectures that can learn feature representation of complex data. This has been the base for most object detection methods developed in the last two decades. For a more comprehensive presentation of object detection methods, we will first present early proposed methods and then the recent advances that use DNNs.

II.3.1 Early Work

The first proposed methods made a number of assumptions about the world to simplify the task. In this simplified world, objects are restricted to polyhedral shapes on a uniform background. Polyhedral shapes have simple geometric representations and it is easy to project them into 2D images where the detection takes place. After transforming the real world into a 2D image, lines are projected into lines and polyhedral faces are projected into polygons. An extended work into projections for object detection was that of L. G. Roberts [95]. In this work he implemented programs for line detection methods as well as line-fitting when parts of the edges are obstructed by other objects. Later similar polyhedral assumptions were used in [96–99].

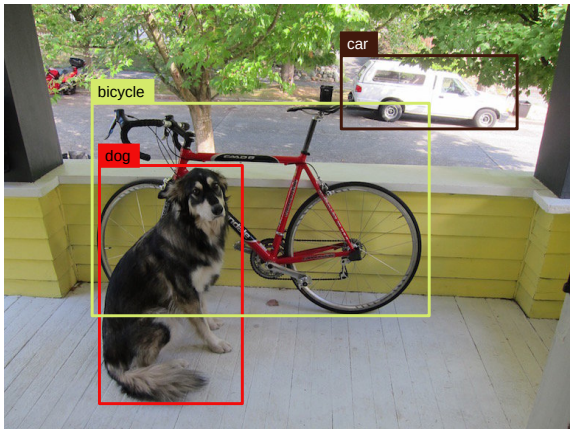


Figure II.2: Object detection example from ImageNet dataset [1].

The main limitations of these methods is the inability to work with curved objects, the requirement of a uniform background and the challenges that arise from shadows. Because of the large number of assumptions being made, these early methods was really hard to work on real world scenes but was the base for the development of more complex detectors.

A significant improvement on the prior methods that overcome many of the previous limitations is the use of multiple views to compute a unique 3D object representation invariant to affine transformations. According to Ullman's theorem, "for rigid transformations, a unique metrical reconstruction is known to be possible from three orthographic views of four points" [100]. A complete representation that describes an object uniquely is not easy to be computed. However representations that achieve rigid and affine invariance are easier to be computed by solving a system of linear equations. The drawback of such simplification is more false positive matches, since affine representations are not unique. In [101] the authors describe the computation of affine representations and describe a method for storing the representations of known objects in hash-tables for real-time detection. In [102] the object representations are computed as a linear combination of 2D images of an object and achieve rigid invariance. Similar to this, in [103] the authors present a method to automatically acquire the representation of an object from noisy image sequences. Their method is also incremental making it possible to process images independently rather than in batches like in the previous methods.

A very significant breakthrough on these geometry-based methods was the development of the Scale Invariant Feature Transform (SIFT) [104]. Unlike the previous methods that compute global representations of objects, SIFT computes local representations that are invariable to image scaling, translation, and rotation, and partially invariant to illumination changes and affine or 3D projection. Very important are the real-time capabilities of SIFT. SIFT representations are usually high-dimensional making the task of finding the nearest neighbors, or the most similar objects in a training set, computationally inefficient. The authors

proposed a modification of the k-d tree algorithm called the best-bin-first search that find the nearest neighbors with high probability but less computations [105]. After the success of SIFT many researchers focused on methods for local representations. The most remarkable of the proposed methods are the Local Binary Patterns (LBP) [106], Haar-like features [107], Shape Contexts [108], SIFT [109], Histogram of Gradients (HOG) [110] and region covariances [111].

Statistical classifiers succeeded the geometry-based methods. A method for training an SVM classifier for face detection was presented in [112]. SVMs were already an established way for classification tasks [67, 113–115]. However, training an SVM on large datasets is a very difficult problem. The training dataset had 50000 images each represented in $10^2 - 10^3$ dimensions. In their work, they propose an optimization method based on the observation that the number of support vectors are very small. Unlike SVMs, AdaBoost [116] training process selects only those features known to improve the predictive power of the model, reducing dimensionality and potentially improving execution time as irrelevant features do not need to be computed. Significant implementations in face detection problems are [107, 117]. Object feature representations are usually large. In both of these implementations the authors choose the most critical features to achieve real-time executions without degrading the detection performance. Neural networks (NNs) are easier to be trained in larger datasets and got a lot of attention either in the form of Multilayer Perceptron (MLP) [118] in [119], Probabilistic Desicion-Based Neural Network (PDBNN) in [120], Convolutional Neural Networks (CNNs) in [121, 122]. These approaches worked on optimizations to reduce the execution time with most impressive being [121] as their method was operating at a fraction of video rates, 5-10 images per second. The NN methods were successful in a limited number of applications, mostly in face detection. Later, the development of deeper architectures of NNs led to accurate and real-time detection methods in more complex tasks.

II.3.2 Deep Learning Approaches

CNNs were a very common choice as an LEC in the 1990s, like in [123], but their use started to decline the next years with the increased use of SVMs. However, their potential was shown in [124] when the authors proposed a deep neural network architecture based on a CNN, they called AlexNet, for the ImageNet LSVRC-2012 contest where it achieved error rate of 15.3% and it was the first time any contestant had a sub-25% error rate. Following this success, researchers tried to understand to what extend deep CNNs can be used for object detection. The answer came with the development of Regions with CNN features (R-CNN) [125]. Object detection, unlike image classification, has the extra task of localizing multiple objects in an image. The approach of R-CNN is to generate a large number of category-independent region proposals using Selective Search [126]. Other region proposal methods are [127–129]. Then, the convolutional part of AlexNet [124]

is used to extract a 4096-dimensional feature vector for each proposed region. Since the regions can have any shape and size, they first transform them to constant size 227x227 pixels regardless of their aspect ratio. Each extracted feature is scored with respect to each class using linear SVMs trained on each class separately. After all proposed regions are scored, a region is rejected if it has an intersection-over-union (IoU) overlap with a higher scoring selected region larger than a learned threshold. The presented R-CNN architecture has a number of optimization properties. There is only one CNN pre-trained and used on all the proposed regions and the resulted feature are low-dimensional. However, the time needed to compute the region proposals and their features is 13s per image on a GPU or 53s per image on a CPU making it not practical in real-time applications. This is mainly due to the fact that the CNN features are extracted per proposed region without shared computations. The second drawback is that the use of SVMs is usually has expensive memory requirements. Finally, the training is not easy as the pipeline training different components, the CNN and the SVMs separately.

Different object detection frameworks were developed to overcome R-CNN's limitations. The main drawback that made the use of R-CNN inefficient and impractical in real-time is that a CNN needs to extract the features for thousands of warped proposed regions separately. The source of this requirement is the fixed input size of CNNs. However, convolutional layers can accept inputs of any size and the requirement of fixed size input only exists due to the fully-connected layers used for classification. SPPNet [130] is equipped with spatial pyramid pooling [131, 132] to overcome the fixed size input limitation. This is a newly introduced layer architecture for CNNs placed on top of the last convolutional layer to generate fixed-length feature output for variable input sizes. With this addition, SPPNet computes the feature map from the entire image only once, and then pool features on each proposed region to generate fixed-length representations. This leads to a significant improvement in execution time which was reported to be 0.142s on a GPU, 102x speedup in comparison to R-CNN on the Pascal VOC 2007 dataset [133] making the use of SPPNet in real-time applications possible. However, this method still involves a multi-stage pipeline with the drawbacks mentioned in the case of R-CNN.

Two years after the development of R-CNN the same author presented Fast R-CNN [134] building on the previous work and addressing some of the drawbacks of R-CNN and SPPNet. A significant contribution of Fast R-CNN is that the training is end-to-end. Similar to SPPnet the feature map from the entire image is extracted only once. Then for each proposed region, a region of interest (RoI) pooling layer extracts a fixed-length feature vector from the feature map. This architecture is a contribution of the same work to deal with proposed regions of arbitrary shapes. This works by dividing the region proposal into equal-sized sections, the desired dimension of the output and finding the largest value in each section. The fixed-size RoI layer output is then connected to fully-connected layers to produce the RoI feature vector. This is connected to two

two sibling networks one that produces softmax probability estimates over K object classes and another layer that outputs four real-valued numbers, describing a bounded-box position, for each of the K object classes. The end-to-end training is achieved by defining a multi-task loss that has one term for the classification loss and one term for the bound box regression loss. Fast R-CNN uses the very deep VGG16 [135] architecture for feature extraction and is 213x faster than R-CNN and 10x faster than SPPNet at test-time and more accurate.

Faster R-CNN [83, 136] was developed to build on the Fast R-CNN architecture and deal with its limitations. Even though Fast R-CNN significantly reduced the execution time for the detection process, it still relies on the very large number of region proposals produced by Selective Search which is its speed bottleneck. The Faster R-CNN introduces the novel *Region Proposal Networks* (RPNs) that is connected to the feature map produced by the CNN in Fast R-CNN. To generate region proposals a small network is slid over the feature map. Each sliding windows produces a low-dimensional feature vector which is fed into two sibling fully-connected layers, a box-regression layer and a box-classification layer. This means that the same fully-connected layers are shared on all spatial locations. The sliding window they use has size 3x3 which corresponds to 228x228 pixel input windows. At each sliding window location they predict region proposals with 3 different scales and 3 different aspect ratios, which they call *anchors*. So for each location, given the $k = 9$ anchors the box-regression layer has $4k$ outputs corresponding to the coordinates of k boxes and $2k$ scores that estimate the probability of object or no object for each proposal. The loss function is similar to multi-task loss of Fast R-CNN and has one term for the binary class box-classification (object or not) and one for the box-regression that is computed using the Intersection-over-Union (IoU) overlap between the predicted regions/anchors and the ground truth bounded boxes. Using the VGG16 model [135], Faster R-CNN has 5 FPS testing execution time on a GPU while improving the detection accuracy over the previous proposed methods on PASCAL VOC 2007 dataset using only 300 proposals per image.

Faster R-CNN was further extended by Mask R-CNN [137] to add image pixelwise segmentation capabilities. Mask R-CNN has exactly the same architecture as Faster R-CNN with the addition of a network branch on top of RPN, and for each Region of Interest (RoI) it predicts a segmentation mask in parallel with the existing branches for box regression and box classification. The mask branch is a small Fully Convolutional Network (FCN) [138, 139] that predicts a segmentation mask in a pixel-to-pixel manner. However the RoI pooling layer was not designed with image segmentation in mind and performs coarse spatial quantization for feature extraction. Mask R-CNN proposes an alternative, quantization-free layer, called *RoIAlign* that preserves the exact spatial locations, to fix the misalignment. The mask branch only adds a small computational overhead and can run at about 200ms per frame on a GPU using a ResNet-101-FPN [140, 141] backbone network. Even though Mask R-CNN can be executed in real-time, the design is not optimized for speed. Trade-offs between speed and accuracy are analysed in [142, 143] by varying image sizes and

proposal numbers. Further, working on speed optimization for the Faster R-CNN architecture, Light Head R-CNN [144] was proposed. This network achieves faster execution by substituting elements of Faster R-CNN with faster alternatives. In their design they use a thin feature map and a tiny Xception like network [145] for the proposals recognition. These modifications achieve 102 FPS testing execution time on the COCO dataset.

The R-CNN and all its extensions are region-based methods, meaning they first compute some region proposals and based on those they compute the bounded boxes and their class. Different methods have been proposed based on a unified framework that directly detect bounded boxes and object classification from full images with a single feed-forward CNN without region proposals. This greatly simplifies training which takes place end-to-end as the whole pipeline is a single network. The simpler design also leads to faster execution times and less computational expensive approaches that can in smaller mobile devices. The most significant approaches based on this framework are YOLO, SSD and CornerNet.

YOLO or You Only Look Once [146] is an object detection algorithm based on a single convolutional network that predicts multiple bounding boxes and the class probabilities for these boxes simultaneously. This approach first divides the input image into an $S \times S$ grid. Each grid cell predicts B bounding boxes and confidence scores for those boxes. The feature map of the input image is generated using a CNN inspired by the GoogleNet [147] and fully connected layers that predict the output probabilities and coordinates. This design is fast because of its simplisity and can run at 45 FPS on a GPU. Using a CNN with 9 layers instead of the24 layers above, it can run at 150 FPS. However, since each cell only predicts a limited number of bounded boxes, 2 in this case, there may be nearby objects in the same cell that will not be detected. Further, YOLO makes more localization errors than Fast and Faster R-CNN, resulting from the coarse division of bounding box location, scale and aspect ratio.

After the success of YOLO different versions were proposed. *YOLOv2* or *YOLO9000* [148] implements various improvements to the YOLO detection method. The custom GoogleNet network is replaced with the simpler DarkNet19 with batch normalization [149] removing the fully connected layers and using boxes of various sizes and aspect ratios learned via k means. YOLOv2 runs faster than the previous YOLO method while its accuracy outperforms Faster R-CNN. *YOLOv3* [150] was proposed to increase the accuracy of the previous methods in the cost of slower execution time at 30 FPS. This has to do with the increase in complexity of the underlying CNN architecture. YOLOv2 used Darknet19, an originally 19-layer network supplemented with 11 more layers for object detection. This architecture often struggled with small object detection. YOLOv3 uses a variant of Darknet53, which originally has 53 layers, by stacking onto it 53 more layers. Moreover YOLOv3 replaces the softmax layer for object classification with a logistic regression on to produce a score on each class. Depending on the score of each class, an object can belong to more than one classes. The recent *YOLOv4* [151] was developed aiming to make training on less powerful single GPUs with

a smaller mini-batch size possible. It is based on the CSPDarknet53 [152] backbone network. The accuracy of the network is improved with the influence of state-of-the-art Bag-of-Freebies and Bag-of-Specials. Bag-of-Freebies refer to methods that make the object detector receive better accuracy without increasing the inference cost. One such method is data augmentation. The purpose of data augmentation is to increase the variability of the input images that will lead to a detection model with higher robustness. This is achieved by augmenting the training set with copies of the existing images on which different transformations are applied like photometric and geometric distortions. Other data augmentation methods try to simulate object occlusion like random erase [153] and CutOut [154]. The choice of Mean Squared Error (MSE) as loss function also belongs to this category. Bag-of-Specials refer to post-processing methods that only increase the inference cost by a small amount but can significantly improve the accuracy of object detection. In this category belong the reception field enhancement [130, 155, 156]. Introducing attention mechanisms [157, 158] also belong to this category. YOLOv4 improves YOLOv3's accuracy and FPS by 10% and 12%, respectively.

SSD [159] was developed right after the original YOLO [146] to preserve real-time speed without sacrificing too much detection accuracy. It is faster than YOLO and its accuracy is slightly better than Faster R-CNN's. To achieve fast detection speed while retaining high detection accuracy, SSD introduces a single-shot detector for multiple categories which is as accurate but faster than the region proposals and pooling of R-CNN and all its extensions. To achieve high detection accuracy, it produces predictions of different scales and aspect ratios from feature maps of different scales, an idea similar to what is used in Faster R-CNN. The additions doesn't affect its end-to-end training simplicity while improve the speed vs accuracy trade-off.

CornerNet [160] differentiates from all the previous object detection methods based on deep learning in the sense that an object bounded box is detected as a pair of key points (top-left and bottom-right). This eliminates the need for designing a set of anchor boxes that has a number of drawbacks. First, a very large set of anchor boxes is needed to ensure sufficient overlap with most ground truth boxes. As a result, only a small fraction of anchor boxes will overlap with ground truth causing a big imbalance between positive and negative examples. Second, as we saw in the previous methods, the use of anchor boxes introduces a number of hyperparameters like the number of boxes, their size and their aspect ratios. The choice of these parameters is based on ad-hoc heuristics and they are hard to be optimized. As a single stage approach, CornerNet uses a single CNN to predict a heatmap for the top-left corners of all instances of the same object category, a heatmap for all bottom-right corners, and an embedding vector for each detected corner which groups the pairs of corners belonging to the same object [161]. The backbone for this stacked architecture is the hourglass network [162]. A novel component of CornerNet is a new type of pooling layer, called *corner pooling*. This serves in helping the CNN better localize corners of bounded boxes. It takes in two feature maps, it max-pools all feature vectors to the right from the first feature map, max-pools all feature vectors

directly below from the second feature map, and then adds the two pooled results together. CornerNet's accuracy outperforms all existing one-stage detectors, except the very recent YOLOv4 [151], with the cost of an execution time of 4 FPS. Its extension, *CenterNet* [163], detects each object as a triplet, rather than a pair, of key points, which improves both precision and recall and the resulting accuracy outperforms all existing one-stage detectors.

CHAPTER III

Inductive Conformal Prediction with Distance Learning

III.1 Introduction

Cyber-Physical systems (CPS) can benefit by incorporating machine learning components that can handle the uncertainty and variability of the real-world. Typical components such as deep neural networks (DNNs) can be used for performing various tasks such as perception of the environment. In autonomous vehicles, for example, perception components aim at making sense of the surroundings like recognizing correctly traffic signs. However, such DNNs introduce new types of hazards that can have disastrous consequences and need to be addressed for engineering trustworthy systems. Although DNNs offer advanced capabilities, they must be complemented by engineering methods and practices that allow effective integration in CPS.

A DNN is designed using learning techniques that require specification of the task, a measure for evaluating how well the task is performed, and experience which typically includes training and testing data. Using the DNN during system operation presents challenges that must be addressed using innovative engineering methods. Perception of the environment is a functionality that is difficult to specify, and typically, specifications are based on examples. DNNs exhibit some nonzero error rate, the true error rate is unknown, and only an estimate from a design-time statistical process is known. Further, DNNs encode information in a complex manner and it is hard to reason about the encoding. Non-transparency is an obstacle to monitoring because it is more difficult to have confidence that the model is operating as intended.

Our objective in this chapter is to complement the prediction of DNNs with a computation of confidence that can be used for decision making. We consider DNNs used for classification in CPS. In addition to the class prediction, we compute set predictors with a given confidence using the conformal prediction framework [23]. We focus on computationally efficient algorithms that can be used for real-time monitoring. An efficient and robust approach must ensure a small and well-calibrated error rate while limiting the number of alarms. This enables the design of monitors which can ensure a bounded small error rate while limiting the number of inputs for which an accurate prediction cannot be made.

Computing well-calibrated confidence is extremely important for designing autonomous systems because accurate measures of confidence are necessary to estimate the risk associated with each decision. The main limitation of existing methods comes from the fact that it is very difficult to select desired confidence values according to the application requirements and ensure bounded error-rate. This is especially important in autonomous CPS applications where decisions can be safety critical. Another important challenge is to

investigate how the computed confidence measures can be used for decision making by autonomous systems and how to handle data for which a confident decision cannot be taken.

The proposed approach is based on conformal prediction (CP) [21, 23]. CP aims at associating reliable measures of confidence with set predictions for problems that include classification and regression. An important feature of the CP framework is the calibration of the obtained confidence values in an online setting which is very promising for real-time monitoring in CPS applications. These methods can be applied for a variety of machine learning algorithms that include DNNs. The main idea is to test if a new input example conforms to the training data set by utilizing a *nonconformity measure* (NCM) which assigns a numerical score indicating how different the input example is from the training data set. The next step is to define a p-value as the fraction of observations that have nonconformity (NC) scores greater than or equal to the NC scores of the training examples which is then used for estimating the confidence of the prediction for the test input. In order to use the approach online, inductive conformal prediction (ICP) has been developed for computational efficiency [23, 24]. In ICP, the training dataset is split into the proper training dataset that is used for learning and a calibration dataset that is used to compute the predictions for given confidence levels. Existing methods rely on NCMs computed using techniques such as k-Nearest Neighbors and Kernel Density Estimation and do not scale for high-dimensional inputs in CPS.

DNNs have the ability to compute layers of representations of the input data which can then be used to distinguish between available classes [39, 79]. In our previous work, we developed an approach for mapping high-dimensional inputs into lower-dimensional representations to make the application of ICP possible for assurance monitoring of CPS in real-time [37]. The approach utilizes the vector of the neuron activations in the penultimate layer of the DNN for a particular input. This low-dimensional representation can be used to compute NC scores efficiently for high-dimensional inputs. In problems where the input data are high-dimensional, such as the classification of traffic sign images in autonomous vehicles, ICP based on these learned embedding representations produces confident predictions. Moreover the execution time and the required memory is significantly lower than using the original inputs and the approach can be used for real-time assurance monitoring of the DNN. The use of low-dimensional learned embedding representations results in improved performance compared with ICP based on the original inputs. However, the underlying DNN is still trained to perform classification and does not learn necessarily optimal representations for computing NC scores.

The main challenge addressed in this chapter is the efficient computation of embedding representations that allows assurance monitoring based on conformal prediction in real-time. The novelty of the approach lies on using distance metric learning to generate representations of the input data and use Euclidean distance as a measure of similarity. Unlike training a classifier where each training input is assigned a ground truth

label and the objective is to minimize a loss function so that the prediction of the classifier will be the same as the label, in distance metric learning, the inputs are considered in pairs. The associated loss function is defined using pairwise constraints such that its minimization will make representations of inputs that belong to the same class be close to each other and representations of inputs belonging to different classes be far from each other. Preliminary results on using appropriate representations for a robotic navigation benchmark with low-dimensional inputs are presented in [164].

The main contribution of this chapter is the leverage of distance metric learning for assurance monitoring of learning-enabled CPS. The presented methods and experimental results have appeared in [165]. The proposed approach based on ICP can be used in real time for high-dimensional data that are typically used in CPS. Different NC functions can be used in ICP to evaluate whether new unknown inputs are similar to the data that have been used for training a learning-enabled component such as a DNN. A NC function assigns a score to a labeled input reflecting how well it conforms to the training dataset. Because the choice of the NC function is very important, the proposed approach utilizes neural network architectures for distance metric learning based on siamese [166] and triplet networks [167] to learn representations and define NC functions based on Euclidean distance. Specifically, the proposed functions compute the NC scores of a new labeled input using (1) the labels of its closest neighbors, (2) how far the closest neighbor of the same class is compared to any other neighbor, and (3) how far the label's centroid is compared to the centroids of the other labels. The main benefit of the approach is that by utilizing distance metric learning in ICP, we reduce the computational requirements without sacrificing accuracy or efficiency.

An important advantage of the approach is that it allows the computation of the optimal significance level that can be used by the assurance monitor to ensure a bounded error rate while limiting the number of inputs for which an accurate prediction cannot be made. Unlike most common machine learning classifiers that assign a single label to an input, ICP computes a set of candidate labels that contains the correct class given a selected significance level. Small significance level values reduce the classification errors but may result in set predictors with multiple candidate labels. In autonomous systems, it is not only important to have predictions with well-calibrated confidence but also to be able to choose the desired significance level based on the application requirements. Even though reducing the number of possible classes may be helpful when the information is provide to a human, in an autonomous system it is desirable that the prediction is unique. Therefore, we assume that set predictions that contain multiple classes lead to a rejection of the input and require human intervention. For this reason, it is desirable to minimize the number of test inputs with multiple predictions. If the prediction is unique, then the monitor ensures a confident prediction with well-calibrated error rate defined by the significance level. If the predicted set contains multiple predictions, the monitor rejects the prediction and raises an alarm. Finally, if the predicted set is empty the monitor

indicates that no label is probable. We distinguish between multiple and no predictions, because they may lead to different action in the system. For example, no prediction may be the result of out-of-distribution inputs while multiple possible predictions may be an indication that the significance level is smaller than the accuracy of the underlying DNN.

This chapter presents a comprehensive empirical evaluation of the approach using three datasets for classification problems in CPS of increasing complexity. The first dataset is the SCITOS-G5 robot navigation dataset [168] for which we use a fully connected feedforward network architecture. The second is a speech recognition dataset which contains audio files of human speech [169]. For this problem, we learn the embedding representations using a DNN with 1D convolutional layers. The third dataset is the German Traffic Sign Recognition Benchmark (GTSRB) [170]. For this dataset, we use a modified version of the VGG16 architecture [171] to learn and generate the embedding representations. We used different combinations of NC functions and distance metric learning architectures and compare them with ICP without distance metric learning. The results demonstrate that that the selected or computed significance levels bound the error-rate in all cases. Moreover, the representations learned by the siamese or triplet networks result in well-formed clusters for different classes and individual training data typically can be captured by their class centroid. Such representations reduce the memory requirements and the execution time overhead while still ensure a bounded small error-rate with limited number of prediction sets containing multiple candidate labels.

The problem definition and the proposed architecture are presented in Section III.3. Sections III.4-III.6 present the details of ICP based on distance learning and assurance monitoring. Finally, we evaluate the performance of our suggested approach on three different applications in Section III.7.

III.2 Related Work

The findings of the state-of-the-art methods described above illustrate the significance of computing well-calibrated and accurate confidence measures. Typically, the main objective is to complement existing machine learning models that are generally unable to produce an accurate estimation of confidence for their predictions with post-processing techniques in order to compute well-calibrated probabilities. An important advantage of such approaches is that they are independent of the underline predictive machine learning models. Therefore, there is no need to redesign and optimize the objective functions used for training which could lead to optimization tasks with high computational complexity.

The proposed work based on ICP produces prediction sets and computes a significance level that will bound the expected error-rate. Similar to existing methods, since the approach is based on ICP, it can be used with any machine learning component without the need of retraining. ICP methods provide very promising results especially when the input data are not very high-dimensional and there are not stringent time

constraints. However, ICP can be impractical when the inputs are, for example, images because of the excessive memory requirements and high execution times. The proposed approach aims to learn appropriate lower-dimensional representations of high-dimensional inputs that make the task of computing confidence measures based on similarities much easier.

III.3 Problem Formulation

A perception component in a CPS aims to observe and interpret the environment in order to provide information for decision making. For example, in autonomous vehicles a DNN can be used to classify traffic signs. The problem is to complement the prediction of the DNN with a computation of confidence. An efficient and robust approach must ensure a small and well-calibrated error rate while limiting the number of alarms to enable real-time monitoring. That is, maximize the autonomous operation time while keeping the error-rate bounded according to the application requirements. Finally, the computation of well-calibrated predictions must be computationally efficient for applications with high-dimensional inputs that require fast decision as, for example, in autonomous vehicles.

During the system operation of a CPS, inputs arrive one by one. After receiving each input, the objective is to compute a valid measure of the confidence of the prediction. The objective is twofold: (1) provide guarantees for the error rate of the prediction and (2) design a monitor which limits the number of input examples for which a confident prediction cannot be made. Such a monitor can be used, for example, by generating warnings that require human intervention.

The conformal prediction framework allows computing set predictors for a given confidence expressed as a significance value [23]. The confidence is generated by comparing how similar a test is to the training data using different nonconformity functions. In our previous work [164] we used DNNs to produce embedding representations for more efficient application of ICP. The additional problem we are solving is the computation of appropriate embedding representations that will lead to more confident decisions. The proposed approach is illustrated in Figure III.1. The main idea is to use distance learning and enable DNNs to learn a lower-dimensional representation for each input on an embedding space where the Euclidean distance between the input representations is a measure of similarity between the original inputs themselves. The ICP approach is applied using the low-dimensional embedding representations and estimates the similarity between a new input and the available data in the training set using a NC function. Using such a representation not only reduces the execution time and the memory requirements but is also more efficient in producing useful predictions. Based on a chosen significance level, ICP generates a set of possible predictions. If the computed set contains a single prediction, the confidence is a well-calibrated and a valid indication of the expected error. If the computed set contains multiple predictions or no predictions, an alarm can be raised to

indicate the need for additional information.

In CPS, it is desirable to minimize the number of alarms while performing the required computations in real-time. Evaluation of the method must be based on metrics that quantify the error rate, the number of alarms, and the computational efficiency. For real-time operation, the time and memory requirements of the monitoring approach must be similar to the computational requirements of the DNNs used in the CPS architecture. Figure III.1 illustrates the proposed architecture for assurance monitoring. At design time, a DNN is trained to produce embedding representations using distance metric learning techniques. Then, NC scores are computed for a labeled calibration set that is not used for training of the DNN. During system operation, the assurance monitor employs the trained DNN to map new sensor inputs to lower-dimensional representations. Using the NC scores of the calibration data, the method produces prediction sets including well-calibrated confidence of the predictions. Ideally, a prediction set should include exactly one class to enable decision making. Alarms can be raised if either the prediction set include multiple possible classes or if it does not contain any.

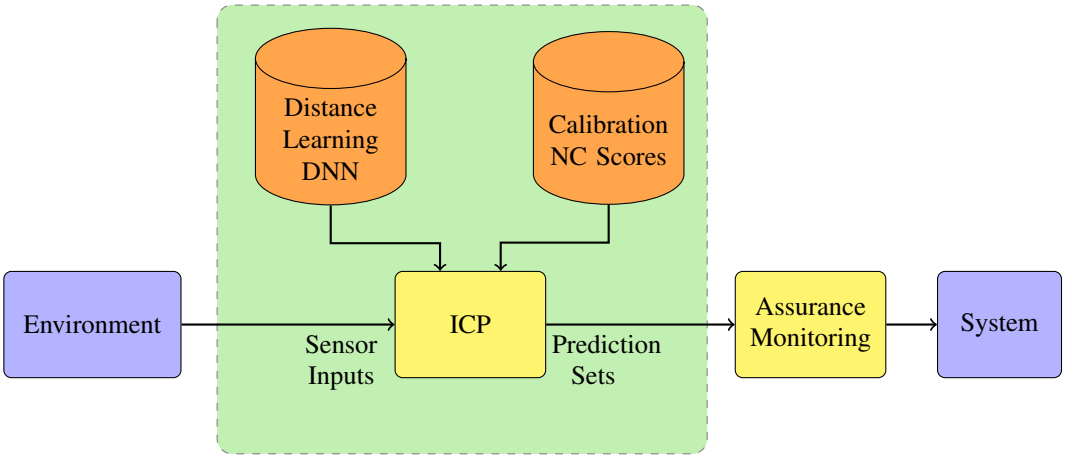


Figure III.1: Assurance monitoring using ICP based on distance learning.

III.4 Distance Learning

The ICP framework requires computing the similarity between the training data and a test input. This can be done efficiently by learning representations of the inputs for which the Euclidean distance is a metric of similarity, meaning that similar inputs will be close to each other as illustrated in Figure III.2. There are different approaches based on DNN architectures that generate embedding representations for distance metric learning.

A *siamese network* is composed using two copies of the same neural network with shared parameters [166] as shown in Figure II.1a. During training, each identical copy of the siamese network is fed with

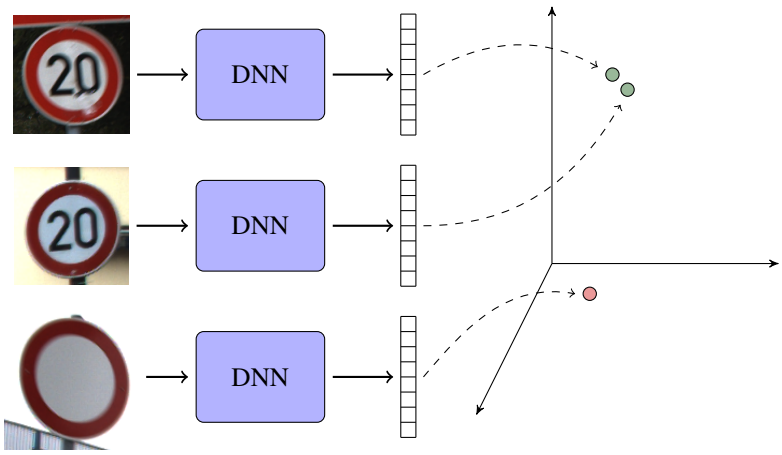


Figure III.2: Embedding representations of input images from the traffic sign recognition dataset.

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:

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

This optimization problem can be solved using the *contrastive loss* function [172]:

$$L(r_1, r_2, y) = y \cdot d(r1, r2) + (1 - y) \max[0, m - d(r_1, r_2)]$$

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 $d(r1, r2) \geq m$, otherwise the parameters of the network are updated to produce more distant representations for those two elements. The reason behind the use of the margin is that when the distance between pairs of different classes are large enough and at most m, there is no reason to update the network to put the representations even further away from each other and instead focus the training on harder examples.

Another architecture trained to produce embedding representations for distance learning is the *triplet network* [167]. A triplet is composed using three copies of the same neural network with shared parameters as shown in Figure II.1b. The training examples consist of three samples, the anchor sample x, the positive

sample x^+ and the negative sample x^-. The samples x and x^+ belong to the same class while x^- belongs to a different class. The embedding representations produced by each network copy will be $r = \text{Net}(x)$, $r^+ = \text{Net}(x^+)$ and $r^- = \text{Net}(x^-)$. The optimization problem described by the Equations III.1 is solved by training the triplet network copies using the *triplet loss* function:

$$L(r, r^+, r^-) = \max[d(r, r^+) - d(r, r^-) + m, 0]$$

.

The margin parameter m separates pairs of different classes by at most m and it is used so that the network parameters will not be updated trying to push a pair even further away when a positive sample is already at least m closer to an anchor than a negative sample. Instead, the training is more efficient when harder triplets are used. The input triplets to the network copies can be sampled randomly from the training data. However, as training progresses it is harder to randomly find triplets that produce $L(r, r^+, r^-) > 0$ that will update the triplet network parameters. This leads to slow training and underfitted models. The training can be improved by carefully mining the training data that produce a large loss [173]. For each training iteration, first, the anchor training data are randomly chosen. For each anchor, the hardest positive sample is chosen, meaning a sample from the same class as the anchor that is located the furthest away from the anchor. Then, the triplets are formed by mining hard negative samples that satisfy $d(r, r^-) < d(r, r^+)$ or semi-hard negatives that satisfy $d(r, r^-) < d(r, r^+) + m$. This way the formed triplet batches will produce gradients to update the shared weights between the DNN copies.

III.5 ICP Based on Distance Learning

We consider a training set $\{z_1, \ldots, z_l\}$ of examples, where each $z_i \in Z$ is a pair (x_i, y_i) with x_i the feature vector and y_i the label of that example. For a given unlabeled input x_{l+1} and a chosen significance level ε, the task is to compute a prediction set Γ^ε for which $P(y_{l+1} \notin \Gamma^\varepsilon) < \varepsilon$, where y_{l+1} the ground truth label of the input x_{l+1}. ICP computes well-calibrated prediction sets with the underlying assumption that all examples (x_i, y_i), $i = 1, 2, \ldots$ are independent and identically distributed (IID) generated from the same but typically unknown probability distribution.

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. For a given test example z_i with candidate label $\tilde{y}_i$, a NC function assigns a numerical score indicating how different the example z_i is from the examples in $\{z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n\}$. There are many possible NC functions that can be used [21–23,36,37]. For example, a NC function can be defined as the number of the k-nearest neighbors to z_{l+1} in the training set

with label different than the candidate label $\tilde{y}_{l+1}$ (k-nearest neighbors NCM). The input space is often high-dimensional which makes storing the whole training set impractical and the computation of the NC scores inefficient. To address this challenge, the proposed approach leverages distance metric learning methods to learn representations that enable applying ICP in real-time.

Nonconformity functions that can be defined in the embedding space learned by siamese and triplet networks are (1) the *k-Nearest Neighbors* (k-NN) [25], (2) the *one Nearest Neighbor* (1-NN) [21], and (3) the *Nearest Centroid* [23]. 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

$$\alpha(x,y) = |i \in \Omega : i \neq y|. \tag{III.2}$$

The 1-NN NCM requires to find the most similar example of a test input x in the training set that is labeled the same as the candidate label y as well as the most similar example in the training set that belongs to any class other than y and is defined as

$$\alpha(x,y) = \frac{\min_{i=1,\ldots,n:y_i=y} d(v,v_i)}{\min_{i=1,\ldots,n:y_i\neq y} d(v,v_i)} \tag{III.3}$$

where $v = f(x)$, $v_i = f(x_i)$, and d is the Euclidean distance metric in the V space.

The Nearest Centroid NCM simplifies the task of computing individual training examples that are similar to a test input when there is a large amount of training data. We expect examples that belong to a particular class to be close to each other in the embedding space 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 NC function is then defined as

$$\alpha(x,y) = \frac{d(\mu_y, v)}{\min_{i=1,\ldots,n:y_i\neq y} d(\mu_{y_i}, v)} \tag{III.4}$$

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 NC score is an indication of how uncommon a test input is compared to the training data. Input data that come from the same distribution as the training data will produce low NC scores and are expected to

lead to more confident classifications while unusual inputs will have higher NC score. However, this measure does not provide clear confidence information by itself but it can be used by comparing it with NCM scores computed using a *calibration set* of known labeled data. Consider the training set $\{z_1,\ldots,z_l\}$. This set is split into two parts, the proper training set $\{z_1,\ldots,z_m\}$ of size $m < l$ that will also be used for the training of the siamese or triplet network and the calibration set $\{z_{m+1},\ldots,z_l\}$ of size $l-m$. The NC scores $a(x_i,y_i)$, $i = m+1,\ldots,l$, of the examples in the calibration set are computed and stored before applying the online monitoring algorithm. Given a test input x with an unknown label y, the method generates a set $|\Gamma^\varepsilon|$ of possible labels $\tilde{y}$ so that $P(y \notin |\Gamma^\varepsilon|) < \varepsilon$. For all the candidate labels $\tilde{y}$, ICP computes the empirical p-value defined as

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

which is the fraction of NC scores of the calibration data that are equal or larger than the NC score of a test input. A candidate label is added to Γ^ε if $p_j(x) > \varepsilon$. It is shown in [23] that the prediction sets computed by ICP are valid, that is the probability of error will not exceed ε for any $\varepsilon \in [0,1]$ for any choice of NC function. Our approach focuses on computing small prediction sets in an efficient manner that allow assurance monitoring approach in real-time.

III.6 Assurance Monitoring

In CPS, it is not only important to have predictions with well-calibrated confidence but also to be able to choose the desired significance level based on the application requirements. ICP computes a prediction set Γ^ε with a chosen significance level ε and Γ^ε may include any subset of all possible classes. Even though reducing the number of possible classes may be helpful when the information is provided to a human, in an autonomous system it is desirable that the prediction is unique, i.e., $|\Gamma^\varepsilon| = 1$. Therefore, we assume that set predictions that contain multiple classes, i.e., $|\Gamma^\varepsilon| > 1$, lead to a rejection of the input and require human intervention. For this reason, it is desirable to minimize the number of test inputs with multiple predictions and we define a monitor with output defined as

$$out = \begin{cases} 0, & \text{if } |\Gamma^\varepsilon| = 0 \\ 1, & \text{if } |\Gamma^\varepsilon| = 1 \\ \text{reject}, & \text{if } |\Gamma^\varepsilon| > 1 \end{cases}.$$

If the set Γ^ε contains a single prediction, the monitor outputs $out = 1$ to indicate a confident prediction with well-calibrated error rate ε. If the predicted set contains multiple predictions, the monitor rejects the

prediction and raises an alarm. Finally, if the predicted set is empty the monitor outputs $out = 0$ to indicate that no label is probable. We distinguish between multiple and no predictions, because they may lead to a different action in the system. For example, no prediction may be the result of out-of-distribution inputs while multiple possible predictions may be an indication that the significance level is smaller than the accuracy of the underlying DNN. Choosing a relatively small significance level that can consistently produce prediction sets with only one class is important. To do this, we apply ICP on the data in the calibration/validation set and compute the smallest significance level ε that does not produce any prediction set with $|\Gamma^{\varepsilon}| > 1$. Assuming the distribution of the test set is the same as the one of the calibration/validation set we expect the same value of ε to minimize the prediction sets with multiple classes on the test data.

The assurance monitoring approach is illustrated by Algorithm 1 and 2. Algorithm 1 shows the tasks that need to be performed at design time where first a distance metric learning network f is trained using the proper training set (X,Y) so that the computed embedding representations will form clusters for each class. Then, using the calibration data, both the NC scores A and the optimal significance level ε are computed and stored. Algorithm 2 shows the tasks that are performed at runtime for a sensor input x_t. The input first needs to be mapped to its embedding representation v_t. Then using the same NC function that is used for the calibration data, we compute the NC scores and the p-values assuming every label j as candidate label. Then the p-values p_j and ε are used to compute the set of candidate labels Γ^{ε}.

Algorithm 1 – Training, Calibration and Significance Level computation.

Require: training data (X,Y), calibration data (X^c,Y^c)
Require: DNN architecture f for distance metric learning
Require: Nonconformity function α
Train f using $(x,y) \in (X,Y)$ ▷ Training
// Compute the representations
$V = f(X)$
$V^c = f(X^c)$
// Compute the nonconformity scores for the calibration data
$A = \{\alpha(v^c,y^c) : (v^c,y^c) \in (V^c,Y^c)\}$ ▷ Calibration
for each v_i^c in $V^c, i = 1..l-m$ **do**
    for each label $j \in 1..n$ **do**
        Compute the nonconformity score $\alpha(v_i^c, j)$
        $p_{ij} = p_j(v_i^c) = \frac{\left|\{\alpha \in A : \alpha \geq \alpha(v_i^c,j)\}\right|}{|A|}$ ▷ empirical p-value
    end for
    Store all p_{ij}
end for
Compute ε such that for each $i \in [1..l-m]$ no more that than 1 of the p-values $p_{ij} \geq \varepsilon$

Algorithm 2 – Assurance Monitoring.

Require: Nonconformity function α and nonconformity scores A
Require: training data or centroids (V, Y) depending on the used nonconformity function α
Require: trained siamese or triplet neural network f for distance metric learning
Require: test input $z_t = (x_t, y_t)$
Require: significance level threshold ε
// Generate prediction sets for each test data x_t
Compute embedding representation $v_t = f(x_t)$
for each label $j \in 1..n$ **do**
    Compute the nonconformity score $\alpha(v_t, j)$
    $p_j(z_t) = \frac{|\{\alpha \in A : \alpha \geq \alpha(z,j)\}|}{|A|}$ ▷ empirical p-value
    if $p_j(z) \geq \varepsilon$ **then**
        Add j to the prediction set Γ^{ε} ▷ Γ^{ε} formation
    end if
end for
if $|\Gamma^{\varepsilon}| = 0$ **then**
    return 0
else if $|\Gamma^{\varepsilon}| = 1$ **then**
    return 1
else
    return Reject
end if

III.7 Evaluation

Our assurance monitor design leverages distance metric learning techniques to compress the input data to lower dimensions in order to make the ICP application more efficient and with lower memory requirements. The objective of the evaluation is to compare how the suggested architecture performs against the baseline ICP approaches as well as investigate the validity/calibration and efficiency (size of set predictions).

III.7.1 Experimental Setup

For the evaluation, we experiment with 3 datasets of variable complexity and input size. First, we use a dataset generated by the SCITOS-G5 mobile robot [168]. This robot is equipped with 24 ultrasound sensors around it that are sampled at a rate of 9 samples per second. Its task is to navigate itself around a room counter-clockwise in close proximity to the walls. The possible actions the robot can take to accomplish this are "Move-Forward", "Sharp-Right-Turn", "Slight-Left-Turn", and "Slight-Right-Turn". The SCITOS-G5 dataset contains 5456 raw values of the ultrasound sensor measurements as well as the decision it took in each sample. Because of the small sensor number, the inputs have one dimension and their size is relatively small. Second, we use a speech recognition dataset which contains 7501 audio samples from speeches of five prominent leaders; Benjamin Netanyahu, Jens Stoltenberg, Julia Gillard, Margaret Thatcher and Nelson Mandela, made available by the American Rhetoric [169]. Each audio sample has 1 second duration, the sampling rate is 16kHz and use Pulse-code modulation (PCM) encoding. Third, the German Traffic Sign

Recognition Benchmark (GTSRB) dataset is a collection of traffic sign images to be classified in 43 classes (each class corresponds to a type of traffic sign) [170]. The dataset has 26640 labeled images of various sizes between 15x15 to 250x250 depending on the distance of the traffic sign to the vehicle. For all datasets we split the available data so that 10% of the samples is used for testing. From the remaining 90% of the data, 80% is used for training and 20% for calibration and/or validation. In the ICP implementations that use the k-NN NC function the number of neighbors k are chosen to be 20, 15 and 40 respectively for the 3 datasets, values that produce stability to outlier data points. The choice of DNN architectures happened according to the complexity of each application so that they will be simple enough to reduce the computational requirements but at the same time achieve good accuracy and data clustering without overfitting. All the experiments run in a desktop computer equipped with Intel(R) Core(TM) i9-9900K CPU, 32 GB RAM and a Geforce RTX 2080 GPU with 8 GB memory.

III.7.2 Baseline

The proposed approach assigns the original inputs into embedding representations for which the Euclidean distance is a measure of similarity between the inputs themselves. In order to understand the effect of the distance metric learning in ICP we compare it with the approaches we used in our previous work [37]. First, the most basic way of applying ICP is using only the original inputs. Then we compare it with the approach we presented in our previous work that uses embedding representations without distance metric learning and this will be the baseline in the following experiments.

The baseline approach computes the embedding representations using the activations of the penultimate layer of a DNN. A DNN is trained as a classifier to predict the class of the input data. The vector of activations of the neurons in the penultimate layer will be considered as the embedding representation of the input. In Figure III.3 there is an illustration of how the embedding representations are generated in the baseline using a DNN with 4 input neurons that classify inputs to two possible classes. The embedding representations are generated in the penultimate layer and are typically reduced in size compared to the inputs. For an accurate comparison between the baseline and the proposed improvements using either the triplet or the siamese network all of these approaches use the same DNN architecture, meaning the embedding representations will also be of the same size.

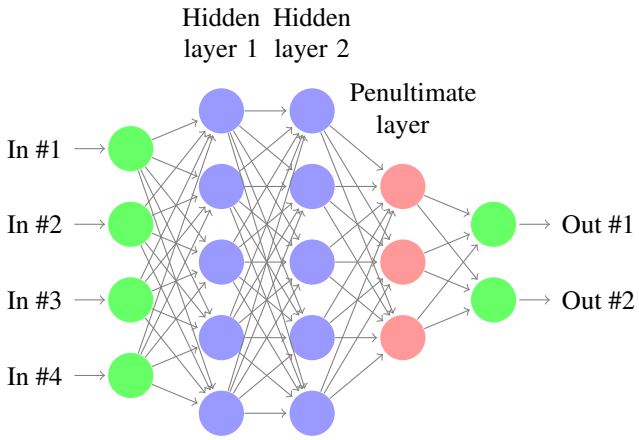


Figure III.3: Baseline DNN architecture

III.7.3 Preprocessing and Distance Learning

The difficulty to compute the NC functions and the memory demands increase as the input size increases. Here we see how the original high-dimensional inputs are mapped to lower-dimensional representations so that the application of ICP will be more efficient as well as the Euclidean distance between two embedding representations is a metric of similarity, property useful in the computation of the NC scores. We evaluate how the use of the embedding representations affect the application of ICP when it is applied on datasets of increasing complexity. First, the input data to the SCITOS-G5 mobile robot is a vector of 24 values. We use a fully connected feedforward DNN to generate embedding representations with size 8. The DNN is trained in either a siamese or triplet network for distance metric learning. The triplet network is trained without mining since this is a small dataset. Second, the speech recognition dataset contain audio samples with duration 1 second. For each audio sample, we add different kind of noises like dishwasher, running tap and exercise bike on half the volume of the speech sample. Then we use FFT to convert the audio samples to their frequency domain. The sampling rate of the speech files is 16kHz, so in the frequency domain it has 8000 components according to the Nyquist–Shannon sampling theorem [174]. A convolutional DNN is used to generate embedding representations of each audio wave in the frequency domain with size 32. In the case when the triplet architecture is used for the DNN's training, the semi-hard negatives mining produce the best results. Finally, the GTSRB dataset contains traffic sign images of variable sizes. In order to be able to use a single DNN to produce embedding representations for the image data, every image is either up-sampled by interpolation or down-sampled to 96x96x3. A convolutional DNN is used to generate embedding representations with size 128. In the triplet case, the training produced better results when mining for hard negatives is used.

We first look at how well the distance metric learning methods cluster data of each class. A commonly

used metric of the separation between classes is the *Silhouette* [175]. For each sample, we first compute the mean distance between i and all other data points in the same cluster in the embedding space

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j) \ .$$

Then we compute the smallest mean distance from i to all the data points in any other cluster

$$b(i) = \min_{k \notin i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j) \ .$$

The silhouette value is defined as

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \ .$$

Each sample i in the embedding space is assigned a silhouette value $-1 \leq s(i) \leq 1$ depending on how close and how far it is to samples belonging to the same and different classes respectively. The closer $s(i)$ is to 1, the closer the sample is to samples of the same class and further from samples belonging to other classes. To compare the representations learned using the different methods as well as compute how much the clustering improves over the original inputs, we compute the mean silhouette over the training data and the validation data separately. In Table III.1, we see that the representations learned by either the siamese or the triplet network form well-defined clusters and are improved over the baseline clusters. On the other hand, the original inputs are not arranged in clusters.

		Training Silhouette	Validation Silhouette
SCITOS-G5	Triplet Embeddings	0.72	0.64
	Siamese Embeddings	0.94	0.8
	Baseline Embeddings	0.23	0.23
	Original Inputs	-0.03	-0.03
Speaker Recognition	Triplet Embeddings	0.64	0.58
	Siamese Embeddings	0.8	0.66
	Baseline Embeddings	0.19	0.19
	Original Inputs	-0.03	-0.03
GTSRB	Triplet Embeddings	0.43	0.43
	Siamese Embeddings	0.75	0.72
	Baseline Embeddings	0.23	0.24
	Original Inputs	-0.22	-0.23

Table III.1: Clustering comparison using the silhouette coefficient

III.7.4 Selecting the Significance Level

First, we illustrate the assurance monitoring algorithm with a test example from the GSTRB dataset. The left side of the Figure III.4 shows the image of a 60km/h speed limit sign. Using nearest centroid as the NC func-

tion and the siamese network, Algorithm 2 can be used to generate sets of possible predicted labels. In the following, we vary the significance level ε and we report the set predictions. When $\varepsilon \in [0.001, 0.004)$, the possible labels are 'Speed limit 50km/h', 'Speed limit 60km/h', 'Speed limit 80km/h'; when $\varepsilon \in [0.004, 0.006)$, the possible labels are 'Speed limit 50km/h', 'Speed limit 60km/h'; and finally when $\varepsilon \in [0.006, 0.0124]$, the algorithm produces a single prediction 'Speed limit 60km/h' which is obviously correct.

Figure III.4: Illustrative example

For monitoring of CPS, one can either choose ε to be small enough given the system requirements or compute ε to minimize the number of multiple predictions. Since the number of multiple predictions decreases when ε increases, we can select ε as the smallest value that eliminates multiple predictions for a calibration/validation set. This can be seen in Figure III.5 where for each dataset, the optimal ε is selected as the significance level value where the performance curve goes to 0. The nearest centroid NC function is used for the plots in this figure.

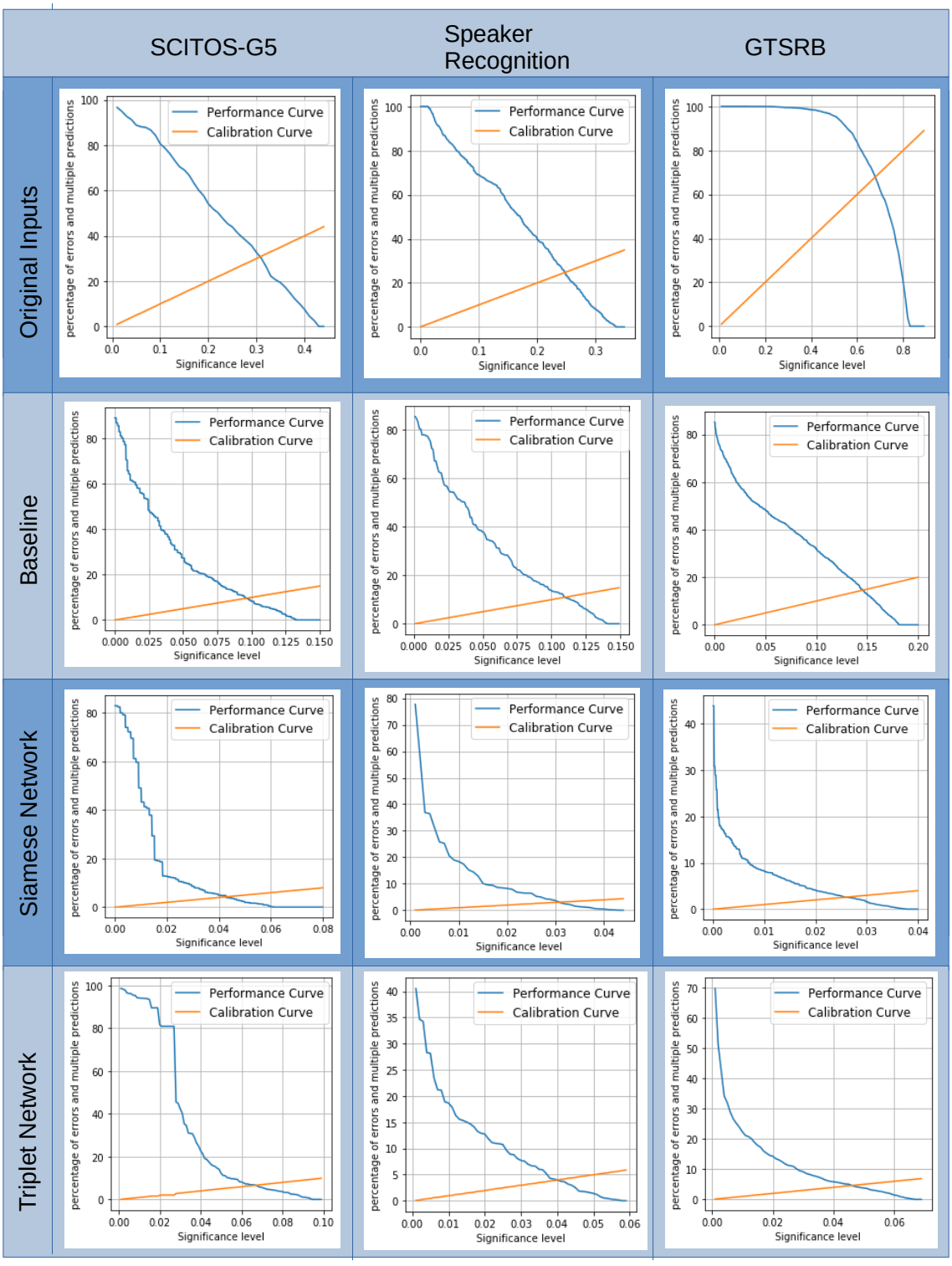


Figure III.5: Performance and calibration curves formed using the validation data from the different datasets using the nearest centroid NC function

Table III.2 shows the results for the different datasets and the various NC functions. First using the cal-

ibration/validation dataset, we select ε to eliminate sets of multiple predictions and we report the errors in the predictions for the testing dataset. The algorithm successfully did not generate any set with multiple predictions for the testing datasets for any of the NC functions other than the 1-NN when it was used in the SCITOS-G5 dataset with representations computed with the triplet network. In this particular case there was no ε that could eliminate the prediction sets with multiple classes and even when $\varepsilon = 1$, 38.6% of the test inputs produced prediction sets with multiple classes. The error-rates are well-calibrated and bounded by the computed or the chosen significance level. One way to compare the different NCMs is by looking at the significance level that is required for ICP to make single predictions. The use of embedding representations could always produce single predictions using significance levels much lower than when the original inputs are used. The significance of the distance metric learning techniques is apparent in the case of the nearest centroid NCM on all the datasets. This is an appealing NCM for its simplicity and the reduced memory requirements. When used as part of the baseline the performance was not as good as the more expensive NCMs. However, leveraging the better clustering that distance metric learning methods achieve, the nearest centroid NCM performs as well or better than the rest of the NCMs on making predictions with low significance level while retaining the computational efficiency. We also evaluate how well the different approaches bound the error-rate for two different values of the significance level. The errors are bounded in most cases no matter if embedding representations are used or not. The percentage of set predictions on the test data that have multiple candidate classes tend to increase the lower the chosen ε is compared to the estimated optimal ε.

			Estimate ε		ε=0.01		ε=0.02	
Dataset	Architecture	NC Function	ε	Errors	Errors	Multiples	Errors	Multiples
SCITOS-G5	Triplet	k-NN	0.087	9.2%	0%	100%	0%	100%
		1-NN	1.0	61.4%	0.4%	88.3%	1%	71.1%
		Nearest Centroid	0.095	8.4%	0%	96.2%	0.5%	84.6%
	Siamese	k-NN	0.066	6.6%	0%	100%	0%	100%
		1-NN	0.078	8.2%	0.4%	71.8%	2.2%	21.4%
		Nearest Centroid	0.062	7.1%	0.2%	45.8%	1.5%	13.4%
	Baseline	k-NN	0.093	7.9%	0%	100%	0.7%	36.3%
		1-NN	0.074	7.5%	0.9%	35.7%	1.3%	27.8%
		Nearest Centroid	0.133	16.1%	0.7%	67.9%	1.6%	55.9%
	Original Inputs	k-NN	0.198	22.3%	0.7%	72.1%	1.6%	58.4%
		1-NN	0.122	12.6%	1%	57.9%	3.7%	37.5%
		Nearest Centroid	0.428	43.5%	0.5%	96.9%	0.7%	95.8%
Speaker Recognition	Triplet	k-NN	0.058	5.2%	1.5%	16.6%	2.5%	10%
		1-NN	0.063	5.9%	2.1%	22.6%	2.8%	14.1%
		Nearest Centroid	0.058	6.3%	1.9%	20.1%	2.5%	14%
	Siamese	k-NN	0.041	3.6%	0%	100%	2.3%	7.3%
		1-NN	0.045	4.5%	0.9%	16.1%	2%	7.6%
		Nearest Centroid	0.043	4%	1.5%	14.8%	1.9%	7.1%
	Baseline	k-NN	0.03	3.1%	0.9%	6.7%	1.6%	2.5%
		1-NN	0.033	3.6%	0.9%	6.9%	2.3%	2.9%
		Nearest Centroid	0.141	14.6%	0.5%	78.2%	2.1%	61.9%
	Original Inputs	k-NN	0.295	29.6%	0%	100%	0%	100%
		1-NN	0.231	22.9%	0.7%	84.82%	1.3%	76.7%
		Nearest Centroid	0.336	33.3	0.7%	100%	1.6%	98.3%
GTSRB	Triplet	k-NN	0.04	4.8%	1.5%	9.3%	2.6%	4.6%
		1-NN	0.031	3.8%	1.4%	8.2%	2.7%	2.9%
		Nearest Centroid	0.067	6.3%	1.5%	22.6%	2.5%	13.9%
	Siamese	k-NN	0.031	3.1%	0.9%	7.4%	1.8%	3.4%
		1-NN	0.035	3.3%	0.8%	9.5%	1.5%	4.8%
		Nearest Centroid	0.038	3.8%	1.2%	8.7%	2.5%	4.1%
	Baseline	k-NN	0.011	1.1%	1%	0.1%	2%	0%
		1-NN	0.003	0.3%	1%	0%	2.1%	0%
		Nearest Centroid	0.182	19.4%	0.9%	71.8%	1.9%	62.8%
	Original Inputs	k-NN	0.731	71.7%	1.2%	98.5%	1.2%	98.5%
		1-NN	—	—	—	—	—	—
		Nearest Centroid	0.824	82.7%	0.9%	100%	2%	100%

Table III.2: ICP performance for the different configurations

III.7.5 Computational Efficiency

In order to evaluate if the approach can be used for real-time monitoring of CPS, we measure the execution times and the memory requirements. Different NC functions lead to different execution times and memory requirements. We compare the average execution time over the testing datasets required for generating a prediction set after the model receives a new test input in Table III.3. The 1-NN NC function on the input space of the GTSRB dataset has excessive memory requirements. Below we present the computational requirements for each NC function and explain the higher requirements of the 1-NN function in more detail.

Dataset	Architecture	NC Function	Execution Time	Memory
SCITOS-G5	Triplet	k-NN	0.2ms	700.6 kB
		1-NN	1.8ms	2 MB
		Nearest Centroid	56μs	324.8 kB
	Siamese	k-NN	0.2ms	700.6 kB
		1-NN	1.6ms	2 MB
		Nearest Centroid	56μs	324.8 kB
	Baseline	k-NN	0.2ms	700.6 kB
		1-NN	1.6ms	2 MB
		Nearest Centroid	58μs	324.8 kB
	Original Inputs	k-NN	0.3ms	2.4 MB
		1-NN	1.9ms	4.1 MB
		Nearest Centroid	59μs	763.1 kB
Speaker Recognition	Triplet	k-NN	0.2ms	15.3 MB
		1-NN	2.1ms	24.6 MB
		Nearest Centroid	74μs	13.1 MB
	Siamese	k-NN	0.2ms	15.3 MB
		1-NN	2ms	24.6 MB
		Nearest Centroid	0.1ms	13.1 MB
	Baseline	k-NN	0.3ms	15.3 MB
		1-NN	2.3ms	24.6 MB
		Nearest Centroid	71μs	13.1 MB
	Original Inputs	k-NN	53ms	723.9 MB
		1-NN	274ms	2.3 GB
		Nearest Centroid	0.1ms	378.7 MB
GTSRB	Triplet	k-NN	0.6ms	70.1 MB
		1-NN	24.6ms	1.4 GB
		Nearest Centroid	0.7ms	38.4 MB
	Siamese	k-NN	0.5ms	70.1 MB
		1-NN	21.8ms	1.4 GB
		Nearest Centroid	0.6ms	38.4 MB
	Baseline	k-NN	0.6ms	70.1 MB
		1-NN	24.4ms	1.4 GB
		Nearest Centroid	0.7ms	38.4 MB
	Original Inputs	k-NN	654ms	8.9 GB
		1-NN	—	—
		Nearest Centroid	4.8ms	2.1 GB

Table III.3: Execution times and memory requirements

Table III.3 reports the average execution time for each test input and the required memory space using different NC functions. Datasets with high-dimensional inputs are challenging for applying ICP in real-time and the results demonstrate the impact of the embedding representations use on the execution times. All the NC functions require storing the the calibration NC scores which are used for computing the test p-values online. The DNN weights need to be stored when embedded representations need to be calculated for every new test input. Furthermore each NCM has a different memory overhead. In the k-NN case, the embeddings of the training data are stored in a $k-d$ tree [176] that is used to compute efficiently the k nearest neighbors. This data structure is used both for the k-NN and 1-NN NC functions. In the 1-NN case, it is required to find

the nearest neighbor in the training data for each possible class which is computationally expensive resulting in larger execution time. The nearest centroid NC function requires storing only the centroids for each class and the additional memory required is minimal.

In conclusion, the evaluation results demonstrate that monitoring based on ICP has well-calibrated error rates in all configurations. Further the use of embedding representations reduces the computational requirements and can lead to decisions with improved significance level. Using distance metric learning methods the training data form well-defined clusters that is essential in the case of the nearest centroid NCM. This improvement makes it a good NCM option for all of the used datasets as it performs as well as the other NCMs but with significantly less computational requirements.

III.8 Concluding Remarks

Cyber-physical systems (CPS) incorporate machine learning components such as DNNs for performing various tasks such as perception of the environment. When used for safety-critical applications they need to satisfy specific requirements that are defined taking into account the acceptable risk and its cost for incorrect decisions. Although DNNs offer advanced capabilities on the decision making process, they cannot provide guarantees on the estimated error-rate. To achieve this they must be complemented by engineering methods and practices that allow effective integration in CPS where an accurate estimate of confidence is needed.

This chapter considers the problem of complementing the prediction of DNNs with a well-calibrated confidence. For classification tasks, the inductive conformal prediction framework allows selecting the significance level according to the requirements of each application. This is a parameter that defines the acceptable error-rate and is a trade-off between errors and alarms. We presented computationally efficient algorithms based on representations learned by underlying DNN models that make possible for ICP to be used for real-time monitoring. The proposed approach was evaluated on three different benchmarks of increasing complexity from a mobile robot with ultrasound sensors, to speaker recognition and traffic sign recognition. The evaluation results demonstrate that monitoring based on the inductive conformal prediction framework using embedding representations instead of the original inputs has well-calibrated error-rates and can minimize the number of alarms when a confident decision cannot be made. When appropriate embedding representations are computed using distance metric learning methods input data that belong to the same class form well-defined clusters. This property is very important when the similarity of a test input to the test data is estimated. That way the training set can be efficiently represented by the centroids of each class which reduces the computational requirements without any loss in performance when compared to the more computationally expensive approaches.

During the experiments we identified a number of challenges that can lead to poor performance of the

proposed method. First, when the datasets are imbalanced both the siamese and the triplet architectures may not learn embedding representations that cluster the under-represented classes well. This affects the efficiency of the NC functions. Second, the training of the triplet networks require mining of training data that will form triplets that lead to large gradients for minimizing the triplet loss function. There is ongoing research for mining algorithms for faster training. One open question for future research is how to utilize all the candidate decisions in the prediction set to deal with the cases when a confident decision cannot be made that will satisfy the significance-level requirements.

CHAPTER IV

Improving Prediction Confidence Using Sequential Sensor Measurements

IV.1 Introduction

Autonomous systems are equipped with sensors to observe the environment and take control decisions. Such systems can benefit from methods that allow to improve prediction and decision making through a feedback loop that queries the sensor inputs when more information is needed [177]. Such a paradigm has been used in a variety of applications such as multimedia context assessment [178], aerial vehicle tracking [179], automatic target recognition [180], self-aware aerospace vehicles [181], and smart cities [182]. In particular, autonomous systems can utilize perception learning-enabled components (LECs) to observe the environment and make predictions used for decision making and control. LECs such as deep neural networks (DNNs) can generalize well on test data that come from the same distribution as the training data and their predictions can be trusted. However, during the system operation the input data may be different than the training data resulting to large prediction errors. An approach to address this challenge is to quantify the uncertainty of the prediction and query the sensors for additional inputs in order to improve the confidence of the prediction. The approach must be computationally efficient so it can be executed in real-time for closing the loop with the system.

Computing a confidence measure along with the model's predictions is essential in safety critical applications where we need to take into account the cost of errors and decide about the acceptable error rate. Neural networks for classification typically have a softmax layer to produce probability-like outputs. However, these probabilities cannot be used reliably as they tend to be too high, they are overconfident, even for inputs coming from the training distribution [4]. The softmax probabilities can be calibrated to be closer to the actual probabilities scaling them with factors computed from the training data. Different methods that have been proposed to compute scaling factors include temperature scaling [4], Platt scaling [12], and isotonic regression [17]. Although such methods can compute well-calibrated confidence values, it is not clear how they can be used for querying the sensors for additional inputs. Conformal prediction (CP) is another framework used to compute set predictions with well-calibrated error bounds [23]. The set predictions can be computed efficiently leveraging a calibration data set [62]. However, such approaches do not scale for high-dimensional inputs such as camera images. In our prior work, we have developed methods handling high-dimensional inputs using inductive conformal prediction (ICP) [37, 164].

This chapter extends our prior work presented in Chapter III by designing a feedback loop between LECs

used for classification and the sensors of an autonomous system in order to improve the confidence of the predictions. The presented methods and experimental results have appeared in [183]. We design a classifier using ICP based on a triplet network architecture in order to learn representations that can be used to quantify the similarity between test and training examples. Given a significance level, the method allows computing confident set predictions. A feedback loop that queries the sensors for a new input is used to further refine the predictions and increase the classification accuracy. The method is computationally efficient, scalable to high-dimensional inputs, and can be executed in a feedback loop with the system in real-time. Using the feedback loop to query the sensors until a confident decision can be made introduces a delay on the decision. The trade-off between accuracy and delay is balanced using a parameter during design-time. The approach is evaluated using a traffic sign recognition dataset. The results show that using multiple sensor readings for a single decision reduces the error rate when compared to decisions based on single measurements. The feedback loop was tuned such that the extra overhead on the time needed for a decision allows for real-time application.

IV.2 Triplet-based ICP

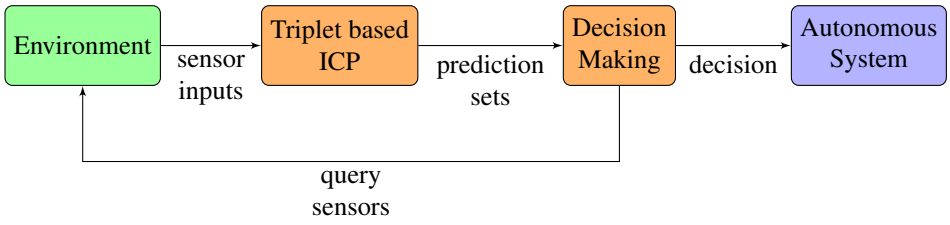


Figure IV.1: Feedback loop between the decision-making process and sensing

We consider an autonomous system that takes actions based on its state in the environment as shown in Figure IV.1. For example, a self-driving vehicle needs to take control actions based on the traffic signs it encounters. We design a classifier using ICP based on a triplet network architecture in order to learn representations that can be used to quantify the similarity between test and training examples. Given a significance level, the method allows computing confident set predictions. A feedback loop that queries the sensors for a new input is used to further refine the predictions and increase the classification accuracy.

The input data in CPS are often high-dimensional which can make ICP inefficient as we saw in the previous chapter. The method presented in this chapter combines ICP with a triplet network to avoid this limitation. Triplet networks are DNN architectures trained to learn representations of the input data for distance learning [167]. The last layer of a triplet network computes a representation of the input. Assuming a trained triplet network and an input x, we write the transformation that takes place by the triplet network as $f(x)$. For training, a triplet network is composed using three copies of the same neural network with shared

parameters. It is trained on batches formed with triplets of data points. Each of these triplets has an anchor data point x, a positive data point x^+ that belong to the same class as x and a negative data point x^- of a different class. The objective is to maximize the distance between inputs of different classes $|f(x) - f(x^-)|$ and minimize the distance of inputs belonging to the same class $|f(x) - f(x^+)|$. To achieve this, training uses the loss function:

$$Loss(x, x^+, x^-) = max(|f(x) - f(x^+)| - |f(x) - f(x^-)| + \alpha, 0)$$

where α is the margin between positive and negative pairs.

The simplest way to form triplets is to randomly sample anchor data points from the training set and augment them by randomly selecting one training sample with the same label as the anchor and one sample with a different label. However, for many of these (x, x^+, x^-) triplets $|f(x) - f(x^-)| >> |f(x) - f(x^+)| + \alpha$, which provides very little information for distance learning and leads to slow training and poor performance. The training can be improved by carefully mining the training data [173]. For each training iteration, first, the anchor training samples are randomly selected. For each anchor, the hardest positive sample is chosen, that is a sample from the same class as the anchor that is located the furthest away from the anchor. Then, the triplets are formed by mining all the hard negative samples, that is the samples that satisfy $|f(x) - f(x^-)| < |f(x) - f(x^+)|$. When the training is completed, only one of the three identical DNN copies is used to map an input x to its embedding representation $f(x)$.

Consider a training set $\{z_1, \ldots, z_l\}$, where each $z_i \in Z$ is a pair (x_i, y_i) with x_i the feature vector and y_i the label. We also consider a test input x_{l+1} which we wish to classify. The underlying assumption of ICP is that all examples (x_i, y_i), $i = 1, 2, \ldots$ are independent and identically distributed (IID) generated from the same but typically unknown probability distribution. For a chosen classification significance level $\varepsilon \in [0, 1]$, ICP generates a set of possible labels Γ^ε for the input x_{l+1} such that $P(y_{l+1} \notin \Gamma^\varepsilon) < \varepsilon$.

ICP uses a function called *nonconformity measure* (NCM), which computes a metric that indicates how different an example z_{l+1} is from the examples of the training set $z_1, \ldots, z_l$. A NCM that can be computed efficiently in real-time is the *k-Nearest Neighbors* (k-NN) [25] defined in the embedding space generated by the triplet network. The k-NN NCM finds the k most similar examples in the training data and counts how many of those are labeled different than the candidate label y of a test input x. We denote $f: X \rightarrow V$ the mapping from the input space X to the embedding space V defined by the triplet's last layer. After the training of the triplet is complete, we compute and store the embeddings $v_i = f(x_i)$ for the training data x_i. Given a test example x with embedding $v = f(x)$, we compute the k-nearest neighbors in V and store their labels in a

multi-set Ω. The k-NN nonconformity of the test example x with a candidate label y is defined as:

$$\alpha(x,y) = |i \in \Omega : i \neq y|$$

For statistical significance testing, p-values are assigned based on the computed NCM scores using a calibration set of labeled data that are not used for training. The training set $(z_1 \dots z_l)$ is split into two parts, the *proper training* set $(z_1 \dots z_m)$ of size $m < l$ that is used for the training of the triplet network and the *calibration set* $(z_{m+1} \dots z_l)$ of size $l - m$ that is used only for the computation of the p-values. The empirical p-value assigned to a possible label j of an input x is defined as the fraction of nonconformity scores of the calibration data that are equal or larger than the nonconformity score of a test input:

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

The p-values are used to form the sets of candidate labels for a given significance level ε. The label j is added to Γ^ε if $p_j(x) > \varepsilon$.

IV.3 Feedback-loop for Querying the Sensors

Only the prediction sets Γ^ε that have exactly one candidate label can directly be used towards the final decision. When $|\Gamma^\varepsilon| \neq 1$ the approach queries the sensors for a new input. Incorrect classifications are more likely to happen during the first time steps of the process as every sensor input offers new information that may lead to a more confident prediction. For example, in the traffic sign recognition task, it is more likely for an incorrect classification to happen when the sign is far away from the vehicle and the image has low resolution as shown in Figure IV.2. To avoid such incorrect classifications, in our method the final decision is made only after k consecutive identical predictions. The parameter k represents a trade-off between robustness and decision time, as larger k leads to additional delay but more confident decisions. Further, very low k values may lead to incorrect decisions while very large values may not allow a timely a decision.

The ICP framework produces well-calibrated prediction sets Γ^ε when inputs are IID. Depending on how small the chosen significance level is, Γ^ε may include a different number of candidate labels. The classification of an input requires $|\Gamma^\varepsilon| = 1$. In our previous work [37, 164], we use a labeled validation set to compute the minimum significance level ε to reduce the prediction sets with more than one candidate label. However, in dynamic systems, sensor measurements change over time. Each new input in a sequence is related to previous inputs and the inputs are not IID. In this case, even though the calculated significance level ε will not lead to $|\Gamma^\varepsilon| > 1$, the actual error rate may not be bounded by ε.

The main idea is to utilize a feedback-loop in order to lower the error-rate. In order to reduce the incorrect predictions that may occur especially for low quality inputs, we require that $|\Gamma^{\varepsilon}| = 1$ with identical single candidate label for k consecutive sensor measurements. When this condition is satisfied for an input sequence, the prediction can be used for decision making by the autonomous system.

Algorithm 3 – Training and Calibration.

Require: training data (X,Y), calibration data (X^c,Y^c)
Require: DNN architecture f for distance metric learning
Require: Nonconformity function α
Train f using $(x,y) \in (X,Y)$ ▷ Training
// Compute the representations
$V = f(X)$
$V^c = f(X^c)$
// Compute the nonconformity scores for the calibration data
$A = \{\alpha(v^c,y^c) : (v^c,y^c) \in (V^c,Y^c)\}$ ▷ Calibration
return A

Algorithm 4 – Runtime Sensor Querying.

Require: Nonconformity function α and calibration nonconformity scores A
Require: training data or centroids (V,Y) depending on the used nonconformity function α
Require: trained triplet neural network f for distance metric learning
Require: sequence of frames S
Require: significance level threshold ε
Require: parameter k for consecutive identical predictions
for each frame $f_i \in S$ **do**
    Query the sensors for a new input x_i
    Compute embedding representation $v_i = f(x_i)$
    for each label $j \in 1..n$ **do**
        Compute the nonconformity score $\alpha(v_i, j)$
        $p_j(z_t) = \frac{|\{\alpha \in A : \alpha \geq \alpha(z,j)\}|}{|A|}$ ▷ empirical p-value
        if $p_j(z) \geq \varepsilon$ **then**
            Add j to the prediction set Γ^{ε} ▷ Γ^{ε} formation
        end if
    end for
    if $|\Gamma^{\varepsilon}| = 1$ **then**
        Frame prediction is Γ_1^{ε}
        if $k-1$ previous frames resulted in the same prediction **then**
            return Γ_1^{ε}
        end if
    end if
end for

IV.4 Evaluation

IV.4.1 Experimental Setup

We apply the proposed method to the German Traffic Sign Recognition Benchmark (GTSRB). A vehicle uses an RGB camera to recognize the traffic signs that are present in its surroundings. The dataset consists of 43

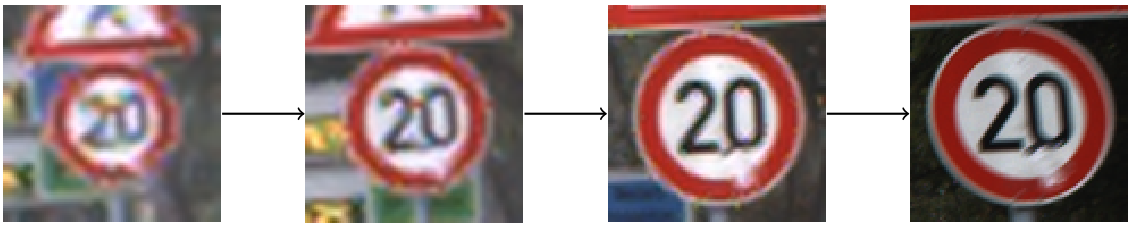


Figure IV.2: Traffic sign over time (in frames)

classes of signs and provides videos with 30 frames as well as individual images. The data are collected in various light conditions and include different artifacts like motion blur. The image resolution depends on how far the sign is from the vehicle as shown in Figure IV.2. Since the input size depends on the distance between the vehicle and the sign, we convert all inputs to size 96x96x3. 10% of the available sequences is randomly sampled to form the sequence test set. 10% of the individual frames is randomly sampled to form another test set. All the remaining frames are shuffled and 80% of them are used for training and 20% are used for calibration and validation.

The triplet network is formed using three identical convolutional DNNs with shared parameters. We use a modified version of the VGG-16 architecture using only the first four blocks because of the reduced input size. A dense layer of 128 units is used to generate the embedding representation of the inputs. 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.

Training Accuracy	Validation Accuracy	IID Testing	Sequence Testing
0.991	0.987	0.986	0.948

Table IV.1: Triplet-based classifier performance

IV.4.2 Model Performance

The triplet network can be used for classification of inputs using a k-Nearest Neighbors classifier in the embedding space. We first investigate how well the triplet network classifier is trained looking at the accuracy of the two test sets. One basic hypothesis of machine learning models is that the training and testing data sets should consist of IID samples. This is confirmed in Table IV.1 where the accuracy for the testing set of IID examples is similar to the training accuracy while the testing accuracy for the set that includes sequences is lower.

In order to investigate which frames are responsible for the larger error-rate in the sequences we plot the average error-rate per frame for the 30 frames of all the test sequences in Figure IV.3. The early frames of each sequence tend to have more incorrect classifications as expected since the sign images have lower

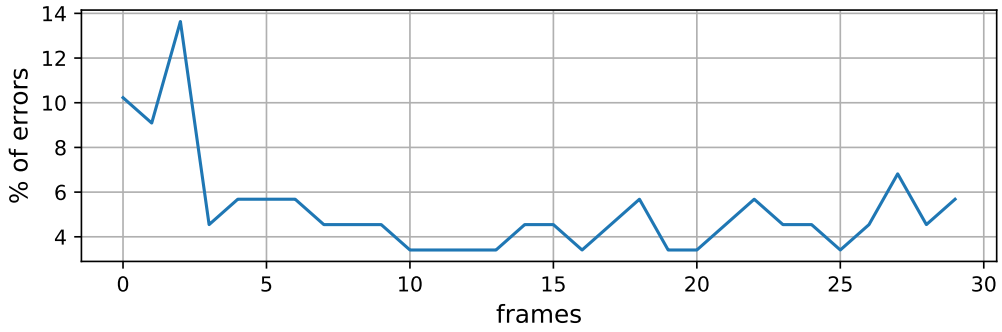


Figure IV.3: Average error per frame for all the test sequences

resolution.

	IID Test		Sequences Test	
ε	Errors	Multiples	Errors	Multiples
0.017	1.7%	0%	5.6%	0%

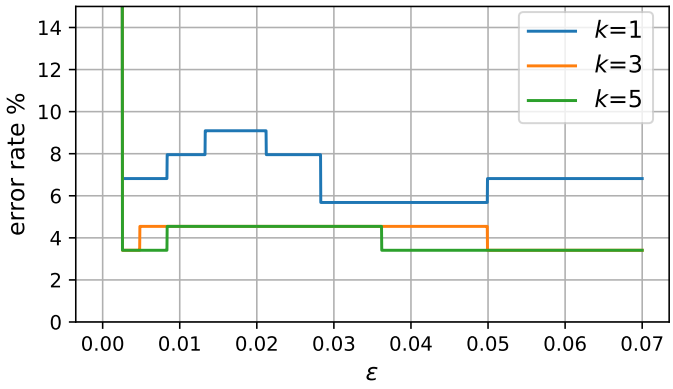
Table IV.2: Triplet-based ICP performance comparison between IID test data and data belonging to sequences

IV.4.3 ICP Performance

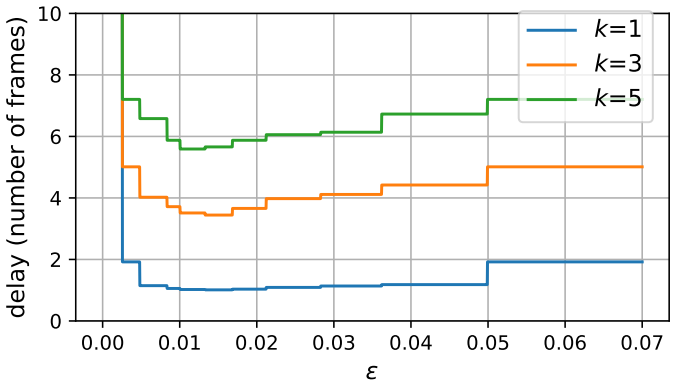
We apply ICP on single inputs to understand how the classifier performs without the feedback loop. The ICP is evaluated for both test sets in Table IV.2. An error corresponds to the case when the ground truth for a sensor input is not in the computed prediction set. We compute the smallest significance level ε that does not produce sets of multiple classes using the validation set. Similar to the point classifier, the ICP classifier produces well-calibrated predictions only for the IID test inputs.

IV.4.4 Improving Prediction Accuracy

We can improve the LEC classification performance using the feedback loop as described in Section IV.3. As we can see in Figure IV.3, the first frames of a sign tend to have more incorrect predictions as they have lower resolution and they lack details. Based on the feedback loop, the LEC uses a new input from the camera until the prediction remains the same for k consecutive frames. Experimenting with different values of k, Figure IV.4 shows that as k increases, the error-rate decreases for most of the ε values but the number of frames required to take a decision increases. When $\varepsilon < 0.003$ the classifier enhanced with the feedback loop could not reach a decision. We also evaluate the efficiency of this classifier regarding to the real-time requirements. A decision for each new sensor query takes on average 1ms, which can be used with typical video frame rates. The memory required to apply the method consists of the memory used to store the



(a)



(b)

Figure IV.4: (a) Error-rate and (b) average number of frames until a decision.

representations of the proper training set and the nonconformity scores of the calibration data (45.9MB) and the memory used to store the triplet network (28.5MB) for a total of 74.4MB.

IV.5 Concluding Remarks

The ICP framework can be used to produce prediction sets that include the correct class with a given confidence. When the inputs to the system are sequential and not IID, applying ICP is not straightforward. Motivated by DDDAS, we design a feedback loop for handling sequential inputs by querying the sensors when a confident prediction cannot be made. The evaluation results demonstrate that when the inputs to the autonomous system are not IID, the error-rate cannot be bounded. However, the addition of the feedback loop can lower the error-rate by classifying a number of consecutive inputs until a confident decision can be made. The running time and memory requirements indicate that this approach can be used in real-time applications.

CHAPTER V

Selective Classification of Sequential Data

V.1 Introduction

Cyber-Physical Systems (CPS) integrate computing, monitoring and control for operation in the physical world. Perception of the environment is a complex process because of the existence of objects that are difficult to model and have complex interaction with the controlled system. Deep Neural Networks (DNN) have the capacity to be trained and generalize their knowledge to make predictions in dynamic environments. CPS can benefit from the integration of DNNs but assurance guarantees are needed that are very challenging to compute. In CPS applications such as autonomous vehicles that needs to recognize traffic signs and take the correct control decisions, the cost of an incorrect classification is much higher than not making any classification when there is no clear distinction between the best prediction and the alternatives. In such a setting, the operation cost over time can be minimized using selective classifiers that evaluate the risk in each classification and either accept the classification or reject it.

Most discriminative machine learning (ML) frameworks make predictions with some notion of confidence under the assumption that the input data are independent and identically distributed (IID) [184]. When there is dependence between the input data this assumption does not hold and the confidence metrics are not accurate. This is an important challenge for CPS to overcome in order to use such frameworks. Sensors in a CPS perceive processes in the physical world that have some duration and individual time instances have some, usually unknown, dependence to previous instances. This leads to miscalibration of confidence estimates and error-rate guarantees are not satisfied [183].

Our approach for improving the confidence of the predictions is based on Inductive Conformal Prediction (ICP) [62]. ICP aims in producing prediction sets that satisfy any error-rate bound guarantees under the IID assumption. The main idea is to test if a new input example conforms to the training data set by utilizing a *nonconformity measure* which assigns a numerical score indicating how different the input example is from the training data set. For any test input, a p-value is assigned to each possible class to decide if a class should be part of the prediction set or not in order to satisfy the chosen error-rate guarantees. This property is valid when the test inputs are IID.

In this chapter we first present how decisions can be made based on p-values computed by ICP. Then, we improve the calibration of the prediction sets computed by ICP and the classification accuracy when the input data are time correlated. We use statistical methods for computing aggregated p-values resulted from

subsequent inputs in a sliding window. We approach the problem as a multiple hypothesis testing problem and show how different combination methods recover ICP validity. The presented methods and experimental results have appeared in [185]. Our main contribution is the design of a selective classifier based on ICP that we call assurance evaluator. This classifier decides if a classification is possible based on the computed p-values for each class. When the highest p-value among all the classes is much higher than the second highest computed p-value we can trust the classification more than cases where at least two of the highest p-values are close to each other. Another contribution of this work 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. This is needed in order to find semantic similarities between data points and handle high-dimensional inputs in real-time. We evaluate the presented approaches on the German Traffic Sign Recognition Benchmark (GTSRB) which provides sequential images of signs as a vehicle moves towards traffic signs.

V.2 Problem

A perception component in a CPS aims to observe and interpret the environment in order to provide information for decision making. In safety-critical systems, predictions on unseen inputs need to have a well-calibrated and bounded error-rate according to predefined safety rules. An efficient and robust approach must ensure a small and well-calibrated error rate while limiting the number of alarms to enable real-time monitoring. Finally, it must be computationally efficient for applications operating on high-dimensional data that require low latency like, for example, in autonomous vehicles.

During the system operation of a CPS, inputs arrive one by one. The inputs may be dependent on each other as shown in Figure IV.2, for example, in a traffic sign recognition system. After receiving each input, the objective is to compute a valid prediction set that satisfies a bound on the error-rate as well as produce classifications based on their trustworthiness. The objective is twofold: (1) provide guarantees for the error-rate of the classification and (2) design an assurance evaluator which minimizes the number of input examples for which a confident prediction cannot be made. The assurance evaluator operates as a selective classifier that generates warnings when no classification can be made and human intervention is needed.

V.3 Selective Classification

ICP computes p-values for each class to construct prediction sets with a chosen significance level. However, its applications are not limited to cases where valid prediction sets are needed. We use multiple hypothesis testing methods to combine the p-values computed on multiple time instances. The aggregate p-values indicate the trustworthiness of each class for particular inputs, over a time horizon, and can be used for point

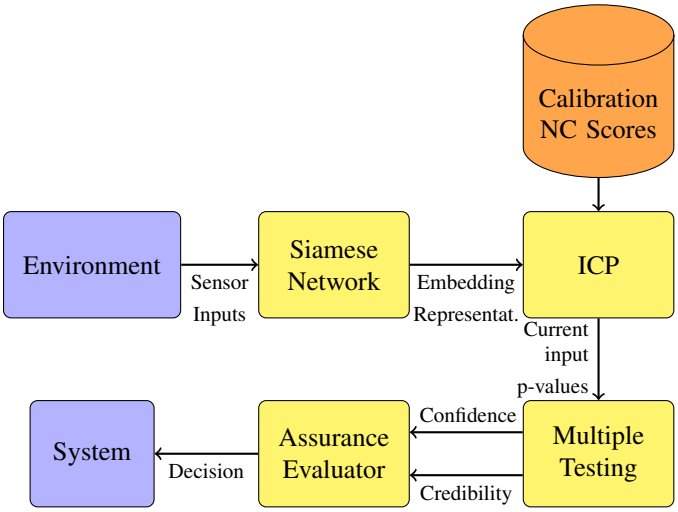


Figure V.1: Execution time architecture

predictions. We define the *credibility* and *confidence* metrics based on the two highest aggregate p-values, $p_{(c)}, p_{(c-1)}$, of all possible classes $p_i, i = 1, \ldots, c$:

$$\text{credibility} = p_{(c)} \tag{V.1}$$

$$\text{confidence} = 1 - p_{(c-1)} \tag{V.2}$$

For a test input x_{l+1} and classification $\hat{y} = \arg\max_{i=1,\ldots,c} p_i$, the credibility shows how credited $\hat{y}$ is and the confidence shows how special it is compared to the other possible labels. These two metrics define the four scenarios shown in Table V.1. The preferred situation is when the largest p-value is close to one and the

Table V.1: Scenarios that can be observed for different values of confidence and credibility.

Credibility	Confidence	Description
High	High	The prefered situation that usually leads into accepting a classification. $p_{\hat{y}}$ is high and much higher than the p-values of the other classes.
High	Low	$p_{\hat{y}}$ is high but there are other high p-values so choosing a single credible class may not be possible.
Low	High	None of the p-values are high for a credible decision.
Low	Low	Not applicable.

rest close to zero. We use an assurance evaluator to decide if a trusted classification can be made and, if not, it will raise an alarm which may require further investigation. For this operation we use the concept of *selective classification* [186, 187]. A selective classifier (f, g) decides whether to keep the classification from

an underlying model or reject it and is defined as:

$$(f,g)(x) \triangleq \begin{cases} f(x), & \text{if } g(x)=1 \\ \text{reject}, & \text{if } g(x)=0 \end{cases} \tag{V.3}$$

where f is the ICP based classifier, and $g : \mathscr{X} \to \{0,1\}$ is a selective function that we call *assurance evaluator*. Consider a function k that evaluates the classifications of f and a threshold θ. The selective function g is defined as, $g_\theta(x|k,f) = \mathbb{1}[k(x,\hat{y}_f(x)|f) > \theta]$. A selective classifier is evaluated using the *coverage* and *risk* metrics. The coverage, $\phi(f,g)$, measures the frequency that the classifications of f are accepted. The risk, $R(f,g)$, is the error-rate in the accepted classifications. These measures can be empirically evaluated using any finite labeled set S_m. The empirical coverage $\hat{\phi}$ and risk $\hat{r}$ are computed as:

$$\hat{\phi}(f,g|S_m) \triangleq \frac{1}{m}\sum_{i=1}^{m} g(x_i) \tag{V.4}$$

$$\hat{r}(f,g|S_m) \triangleq \frac{\frac{1}{m}\sum_{i=1}^{m} l(f(x_i),y_i)g(x_i)}{\hat{\phi}(f,g|S_m)} \tag{V.5}$$

where $l(f(x_i),y_i) = 1$ if $f(x_i) = y_i$ otherwise $l(f(x_i),y_i) = 0$.

For a given classifier f we optimize the assurance evaluator g based on the area under the risk-coverage (RC) curve (AURC) defined in [188]. Consider a set of n labeled points V_n and let the set $\Theta \triangleq \{k(x,\hat{y}_f(x)|f) : (x,y) \in V_n\}$ of threshold values. Using these threshold values to define the selective function g we can compute n empirical risk and coverage values and plot a RC curve. When two assurance evaluators are compared, preferable is the one with lower risk at the same coverage. So a metric for evaluation of pairs (f,g) is the AURC:

$$\text{AURC}(k,f|V_n) = \frac{1}{n}\sum_{\theta\in\Theta} \hat{r}(f,g_\theta|V_n). \tag{V.6}$$

The assurance evaluator is constructed with a choice of a classification evaluator function k and a threshold θ. A function k needs to be chosen to minimize AURC, which intuitively minimizes the average empirical risk. We express k as a linear combination of the credibility and confidence, computed by ICP,

$$\begin{aligned} k(x,\hat{y}_f(x)) = a \cdot \text{credibility}(x,\hat{y}_f(x)) \\ + b \cdot \text{confidence}(x,\hat{y}_f(x)) \end{aligned} \tag{V.7}$$

We compute the optimal parameters a and b that minimize the AURC with a grid search in $[-1,1]$. Based on the RC curve and the application requirements regarding the accepted risk and coverage of the assurance

evaluator (r^*, ϕ^*), a threshold θ is chosen such that $(\hat{r}, \hat{\phi}) = (r^*, \phi^*)$.

V.4 Multiple Testing Of Single Hypothesis

ICP forms prediction sets with theoretical guarantees on the error-rate based on computations of p-values for each class. When the inputs to be classified are composed of sequential data arriving one after the other, it is natural to utilize statistical methods for combining individual p-values to improve the accuracy and efficiency over the individual classifications. We, first, briefly present how ICP computes p-values and prediction sets and in then second part of this section we present different ways of computing aggregate p-values.

V.4.1 Inductive Conformal Prediction

Given a test input x_{l+1}, ICP computes a prediction set Γ^{ε} of labels with enough evidence to be the true label. We consider the more fundamental question: given a test input x_{l+1} belonging in class $y_{l+1} \in Y$, is label $\hat{y}_{l+1} : \hat{y}_{l+1} \in Y$ the true label? Hypothesis testing is a statistical method used to make decisions on whether a hypothesis is true based on a finite number of data. The question to be answered is translated into two competing and non-overlapping hypothesis. (1) The *null hypothesis*, H_0, is the argument believed to be true and (2) 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. Considering again a test input x_{l+1}, the question whether $\hat{y}_{l+1}$ is the true class, is written using the above notation. 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.

The p-value is a measure of how likely it is that the pair $(x_{l+1}, \hat{y}_{l+1})$ has occurred under the null hypothesis, $\hat{y}_{l+1} = y_{l+1}$. It is the probability for this data point to occur or something that is as, or more, extreme. On the assumptions that the null hypothesis is true and that the sampling distribution is given by a probability density function (PDF), the distribution of p-values is uniform in the interval $[0,1]$.

The null distribution is usually unknown in practice and ICP approaches it using a labeled calibration set. First we consider a training set $\{z_1, \ldots, z_l\}$ of examples, where each $z_i \in Z$ is a pair (x_i, y_i) with x_i the feature vector and y_i the label of that example. This set is split into the proper training set $\{z_1, \ldots, z_m\}$ of size $m < l$ and the calibration set $\{z_{m+1}, \ldots, z_l\}$ of size $l-m$. Central to the framework is the use of nonconformity measures (NCM), a metric that indicates how different an example z_{l+1} is from the examples of the training set. Assuming that the test data are generated from the same distribution as the calibration data, the null distribution is the distribution of the nonconformity (NC) scores of the calibration set pairs (x_i, y_i), $i = m+1, \ldots, l$. When the input data are high-dimensional, for example images, computing the Euclidean distance between two inputs is not a proper way to estimate their similarity. The Euclidean distance

does not take into consideration the spatial relationships of pixels so small translations or rotations between similar images may lead to a large distance. We use a *siamese network* to transform the original inputs into lower dimensional embedding representations in a space where the Euclidean distance is a measure of how semantically similar the original inputs are. A siamese network is composed using two copies of the same neural network with shared parameters [166]. More information on this architecture and how it is trained are given in Section II.2.

NCM is a function that computes the dissimilarity between an example z_{l+1} and the examples of the training set $z_1, \ldots, z_l$. There are many different possible NCMs that can be used [21–23, 36, 37, 164]. Having used all the above NCMs in our previous research work, we find the *nearest centroid* to have good trade-off characteristics between the NC scores quality, the memory requirements and the applicability in real-time systems when combined with distance metric learning. The nearest centroid NCM simplifies the task of computing individual training examples that are similar to a test input when there is a large amount of training data. We expect the embedding representations of examples that belong to a particular class to be close 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 NC function is then defined as

$$\alpha(x,y) = \frac{d(\mu_y, v)}{\min_{i=1,\ldots,n:y_i \neq y} d(\mu_{y_i}, v)} \tag{V.8}$$

where v the embedding representation of the feature vector x. It should be noted that for computing the nearest centroid NCM, we need to store only the centroid of each class.

The trustworthiness of a particular class given a test input is expressed as a p-value. It is computed as the fraction of the calibration NC scores that are greater or equal to the NC score computed for the current hypothesis testing. Assume a test input x_{l+1}, $v_{l+1} = f(x_{l+1})$, and we test the null hypothesis, the class y^j is the correct class, $y_{l+1} = y^j$:

$$p_j(x_{l+1}) = \frac{|\{\alpha \in A : \alpha \geq \alpha(x_{l+1}, y^j)\}|}{|A|} \tag{V.9}$$

where A the set of calibration NC scores and $\alpha(x_{l+1}, y^j)$ the NC score of the pair (x_{l+1}, y^j). This hypothesis is accepted if $p_j(x_{l+1}) > \varepsilon$, where ε the significance level. This process is repeated for all possible classes $y^j, j = 1, \ldots, |Y|$. All the classes that were not rejected for a chosen ε are added in the prediction set Γ^ε.

ICP computes well-calibrated prediction sets, defined as $P(y_{l+1} \notin \Gamma^\varepsilon) < \varepsilon$, 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 exchangable [189]. However, the choice of ε is important for the

computation of efficient prediction sets. The best case scenario for Γ^{ε} is $|\Gamma^{\varepsilon}| = 1$ and the only class that satisfies $p_j(x_{l+1}) > \varepsilon$ is the ground truth class.

V.4.2 Combining Multiple p-values

The problem of multiple hypothesis testing appears when a decision about a null hypothesis needs to be made after a number of tests $K > 1$. According to the problem we consider in this paper, the same null hypothesis is tested over K consecutive frames of a sequence. The p-values $p_1, \ldots, p_K$, obtained from the K individual hypothesis tests need to be combined into a single p-value. Since the individual tests take place on consequent frames of a sequence, it is expected the p-values are dependent with each other. For this combination to be used in the ICP framework, the combined p-values during testing should lead to valid prediction sets with error-rate equal to the chosen significance-level.

V.4.2.1 Merging Functions

In [190], authors propose general methods for combining a number of p-values without making any assumptions about their dependence structure. Different functions for combining p-values can be derived from the generalized mean p-value (GMP):

$$M_{r,K}(p_1, \ldots, p_K) = \left(\frac{p_1^r + \cdots + p_K^r}{K} \right)^{1/r} \tag{V.10}$$

where $r \in [-\infty, \infty]$, $K = 2, 3, \ldots,$. Merging functions for combing p-values without the independence assumption are constructed as

$$p_{\text{comb}} = a_{r,K} M_{r,K}(p_1, \ldots, p_K)$$

Special cases of merging functions that we use in our evaluation and are derived from (V.10) are the minimum, maximum, arithmetic mean and geometric mean. When $r \to -\infty$ the resulting merging function is known as the Bonferroni method and one of the most well-known methods for multiple testing:

$$p_{\min} = K \min(p_1, \ldots, p_K) \tag{V.11}$$

Similarly when $r \to \infty$:

$$p_{\max} = \max(p_1, \ldots, p_K) \tag{V.12}$$

(V.11) and (V.12) are generalized in [191] to use order statistics of the p-values:

$$p_{\text{ord}} = \frac{K}{k} p_{(k)} \tag{V.13}$$

where $p_{(k)}$ is the kth smallest p-value of the $p_1, \ldots, p_K$. When $r = 1$ individual p-values are combined using the arithmetic average. However the arithmetic average of a number of p-values is not always a p-value. In [192], Theorem 1, it is shown that multiplying the arithmetic average with a factor of 2 is a p-value.

$$p_{\text{arith\_avg}} = \frac{2}{K} \sum_{i=1}^{K} p_i \tag{V.14}$$

When $r = 0$ individual p-values are combined using the geometric average:

$$p_{\text{geom\_avg}} = e \times \left(\prod_{i=1}^{K} p_i \right)^{1/K} \tag{V.15}$$

V.4.2.2 Quantile Combination Approaches

The general quantile combination approach produces p-values that are uniformly distributed in $[0, 1]$ when the combined p-values are independent. If $X_1, X_2, \ldots, X_n$ samples from a continuous distribution X with CDF F_X, then the samples $U_i = F_X(X_i)$ follow a uniform distribution U with CDF $F_U(u) = u$. The proof is straightforward and is added here for completeness.

$$F_U(u) = \mathbb{P}[U \leq u] = \mathbb{P}[F_X(X) \leq u] = \mathbb{P}\left[X \leq F_X^{-1}(u)\right] = F_X\left(F_X^{-1}(u)\right) = u \tag{V.16}$$

The advantage of leveraging this property is that p-values can be combined using any arbitrary function f and then transform the resulting p-values to a uniform distribution using the CDF of f. However, in our application the p-values computed by ICP on consequent frames are not independent and cannot be considered as independent samples from a continuous distribution. This means that the transformed p-values may not be uniformly distributed affecting the global validity of ICP. Since we do not know the dependence structure between the inputs, these methods could still result in an ICP valid in regions of interest and we experiment with their use in CPS. There is a large number of quantile combination methods proposed in the literature that transform and combine p-values using functions with CDF that can be expressed in closed form or can be computed efficiently. However, because of their independence requirement, this is not an exhaustive list of methods but an evaluation of the most commonly used ones. A number of quantile combination methods is evaluated using ICP computed p-values for multiple underlying model ensembles in [193, 194].

One way of combining multiple p-values is using their product. This is commonly known as the Fisher's method [195]. Assuming that $p_1, p_2, \ldots, p_k$ are samples from a uniform distribution, then

$$h_i = -2 \log p_i$$

follows a chi-squared distribution with 2 degrees of freedom. The sum of independent chi-squared distributions is also a chi-squared distribution with degrees of freedom equal to the sum of the degrees of freedom of the individual chi-squared distributions. The CDF of the chi-squared distribution is expressed in closed form so a sequence of k independent p-values can be combined efficiently as

$$p_{\text{prod}} = \mathbb{P}\left\{ y \leq -2\sum_{i=1}^{K} \log p_i \right\} = t \sum_{i=0}^{K-1} \frac{(-\log t)^i}{i!} \tag{V.17}$$

where $t = \prod_{i=1}^{K} p_i$.

A similar approach is the Stouffer's z-transform [196], which first maps the uniformly distributed and independent p-values to random variables that follow the normal distribution

$$h_i = \Phi^{-1}(1 - p_i)$$

where Φ is the cumulative normal distribution. The random variables $h_i, i = 1, \ldots, K$ are then combined such that

$$h = \frac{\sum_{i=1}^{K} h_i}{\sqrt{K}}.$$

The sequence of p-values, $p_i, i = 1, \ldots, K$, is combined by sampling the CDF of h

$$p_{\text{z}} = \mathbb{P}\left\{ y \leq \frac{\sum_{i=1}^{K} h_i}{\sqrt{K}} \right\} = 1 - \Phi(h) \tag{V.18}$$

which is not in closed form but easily computed by most mathematical software. This method is extended in [197] to assign weights on independent experiments. This can be useful in our applications as more recent inputs may be more significant than older ones. We call it the weighted Stouffer's method:

$$p_{\text{z\_weighted}} = 1 - \Phi\left(\frac{\sum_{i=1}^{K} w_i Z_i}{\sqrt{\sum_{i=1}^{K} w_i^2}} \right) \tag{V.19}$$

The weights we assign are larger for recent inputs in a sliding window and decrease over time so that $w_i = i / \sum_{j=1}^{K} j$.

To keep our evaluation of quantile combination methods consistent with the merging function presented earlier, order statistics functions, like min and max, can be used to produce p-values by sampling their CDF. Let $p_{(r)}$ be the rth smallest among K independent p-values. These p-values follow the $Beta(r, K - r + 1)$ distribution [198]. In this case the CDF is an incomplete beta function. It cannot be expressed in closed form but it is easily computed by most mathematical software.

The Cauchi combination test [199] is more recent and although it is based on the quantile combination methods, it is developed to be applied under arbitrary correlation structures. Assuming that $p_1, p_2, \ldots, p_k$ are samples from a uniform distribution, then the components

$$h_i = \tan\{(0.5 - p_i)\pi\}$$

follow a standard Cauchi distribution. The sum $T = \sum_{i=1}^{K} h_i$ also has a standard Cauchi distribution under the null and the its CDF can be computed in closed form:

$$p_{\text{Cauchi}} = \mathbb{P}\left\{ y \leq \sum_{i=1}^{K} h_i \right\} = \frac{1}{2} - \frac{\arctan T}{\pi} \tag{V.20}$$

Expressed in closed form, similar to the previous methods, it has low computational requirements.

V.4.2.3 Empirical CDF Computation

In practice, p-value transformations using CDFs are not always possible. The reason is twofold: (1) not all arbitrary combination functions have a CDF that can be expressed in closed form and (2) the p-values to be combined may be dependent. Instead of using CDFs we compute an Empirical Cumulative Distribution Function (ECDF) from a set of calibration sequences. We first compute the combined p-values that are consistent with the null hypothesis using any arbitrary law $f(p_1, \ldots, p_K)$. Then the ECDF $F_X(x)$ is computed on the finite set of combined p-values. During test time when a sliding window of K frames is present, the p-values of each class are combined with an arbitrary law and the computed ECDF is used to recover validity of the combined p-values

$$p_{\text{comb}} = F_X\left[f(p_1, \ldots, p_K)\right] \tag{V.21}$$

where F_X is the computed ECDF. To understand the effects of ECDF, during evaluation we use simple combination laws consistent with the CDFs above.

V.5 Evaluation

Our assurance evaluator design leverages distance metric learning techniques to compress the input data to lower dimensions in order to make the ICP application more efficient and with lower memory requirements. The first part of the evaluation compares the performance of our assurance evaluator with a baseline evaluator based on thresholds on the softmax outputs of a DNN classifier. This is a unit test without considering the combination of multiple inputs. The second part of the evaluation extends the assurance evaluator for applications on sequential data and compares our multiple hypothesis testing approach based on ICP with the

baseline snapshot ICP application.

V.5.1 Experimental Setup

We apply the proposed method to the German Traffic Sign Recognition Benchmark (GTSRB). A vehicle uses an RGB camera to recognize the traffic signs that are present in its surroundings. The dataset consists of 43 classes of signs and provides videos of 30 frames as well as individual images that are not part of sequences. The data are collected in various light conditions and include different artifacts like motion blur and obstructions by other objects. The image resolution depends on how far the sign is from the vehicle as shown in Figure IV.2. Since the input size is variable, we convert all inputs to size 30x30x3. 10% of the available sequences is randomly sampled to form the sequences used for testing. From the remaining sequences, 20% is used to compute the ECDFs an the remaining sequences form the training set. The training set is split into the proper training set and the calibration set with a ratio of 5:1. 90% of the individual images that are not part of sequences augment the proper training set and the calibration set with the same ratio as above and the remaining 10% forms another test set so that we can use it as unity test for the baseline ICP.

The siamese network is formed using two identical convolutional DNNs with shared parameters. The architecture we chose to use is the one described in [200] for a similar application. A dense layer of 256 units is used to generate the embedding representation of the inputs. 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.

V.5.2 Siamese Network Evaluation

We first investigate how well the siamese network is trained looking at two separate metrics. One is the classification accuracy. The siamese network can be used for classification of inputs using a k-Nearest Neighbors classifier in the embedding space. One basic hypothesis of machine learning models is that the training and testing data sets should consist of IID samples. This is confirmed in Table V.2 where the accuracy for the test set of IID examples is similar to the training accuracy while the testing accuracy for the set that includes sequences is lower.

Then we evaluate how well the siamese network clusters data of each class. A commonly used metric of the separation between classes is the *silhouette* [175]. For each sample, we first compute the mean distance between i and all other data points in the same cluster in the embedding space

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j) \, .$$

Then we compute the smallest mean distance from i to all the data points in any other cluster

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j) \ .$$

The silhouette value is defined as

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \ .$$

Each sample i in the embedding space is assigned a silhouette value $-1 \leq s(i) \leq 1$ depending on how close it is to samples belonging to the same class and how far it is to samples belonging to different classes. The closer $s(i)$ is to 1, the closer the sample is to samples of the same class and further from samples belonging to other classes. To compare the representations learned in the different sets of data, we compute the mean silhouette value.

Table V.2: Siamese accuracy evaluation

	Accuracy	Silhouette
Training	0.962	0.48
Validation	0.95	0.45
Test IID	0.955	0.44
Test Sequences	0.923	0.51

V.5.3 Softmax Baseline and Selective Classification with Individual Inputs

In most real world application making a classification is not enough. Classifications need to be complemented with estimations of the expected error-rate. In a safety-critical application like the traffic sign recognition the expected error-rate is a metric that is used for decision-making and its accurate estimation is more important than the average accuracy of the classifications. In such applications the use of highly accurate DNN architectures that lack well-calibrated assurance metrics can be unsafe. To understand this problem with commonly used, accurate, architectures we train a DNN classifier with the same architecture as the one used for the DNNs in the siamese network. In the case of the DNN classifier we add a softmax layer with 43 outputs after the embedding layer. The softmax layer outputs for each class a value in $[0,1]$ indicating the certainty in predicting each class. To understand if the softmax values are accurate regarding the predictions on the sequential test data we split the predictions in 25 bins according to the softmax value assigned to them. Then for each of the beans with at least one entry we compute the average softmax value of the predictions as well as the true accuracy of the predictions in the bean. The plot of the accuracy with respect to the softmax output is called reliability diagram and it is shown in Figure V.2. In this plot we see that the softmax output assigned to the DNN's predictions are generally overconfident. The problem is significant in large softmax outputs as

predictions with softmax outputs ≈ 0.94, that could be considered trustworthy, have true accuracy of only 0.66.

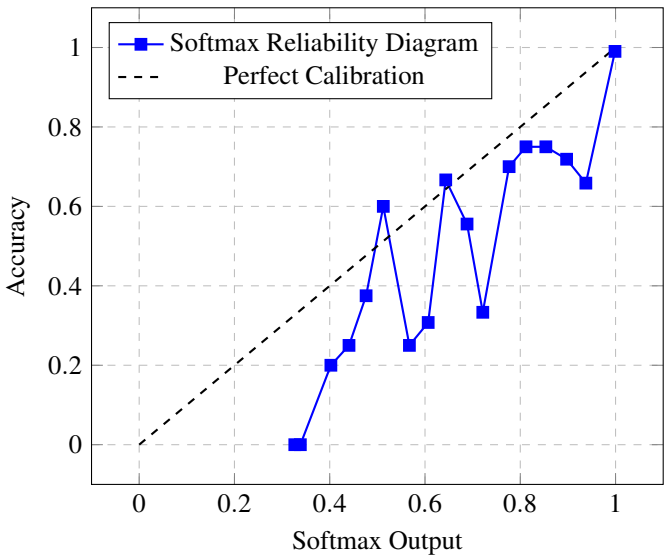


Figure V.2: Reliability diagram of a classifier that uses softmax output for assurance assessment.

To improve the reliability of the decisions we use selective classification based on ICP, that is descrbed in Section V.3. In this part of the evaluation we investigate the performance of the selective classification with respect to the accuracy and calibration when the inputs are considered individually. The siamese network that is evaluated in Section V.5.2 is used to to compute embedding representations of the training and calibration data. We extract the centroids of the training embedding representations use them to compute the nonconformity scores of the calibration data. The decisions are taken based on the p-values computed for all the possible labels of an input using the equations (V.7, V.1, V.2). In order to compute the optimal parameters a and b we perform a grid search to identify the pair that minimizes equation V.6 when computed on the labeled calibration sequential data. The optimal parameters are computed to be $a = 0$ and $b = 0.1$. Then, these parameters are used to form the assurance evaluator that is used to make decisions on the test sequences. In the Figure V.3a we see the value of risk (the error-rate of the accepted decisions) with respect to the chosen threshold for both the calibration and test sequences. Figure V.3b shows the RC curve when the threshold takes all possible values. In order to decide for the threshold parameter we we choose an operation point on the RC curve that produces the best trade-off characteristics between the risk and coverage values. We see that regardless the choice of threshold, the observed risk on the test sequences is bounded by the expected risk computed on the calibration sequences. For example, an operation point with 89% coverage and 6% risk on the calibration sequences, will result in 91% coverage and 4% risk on the test data.

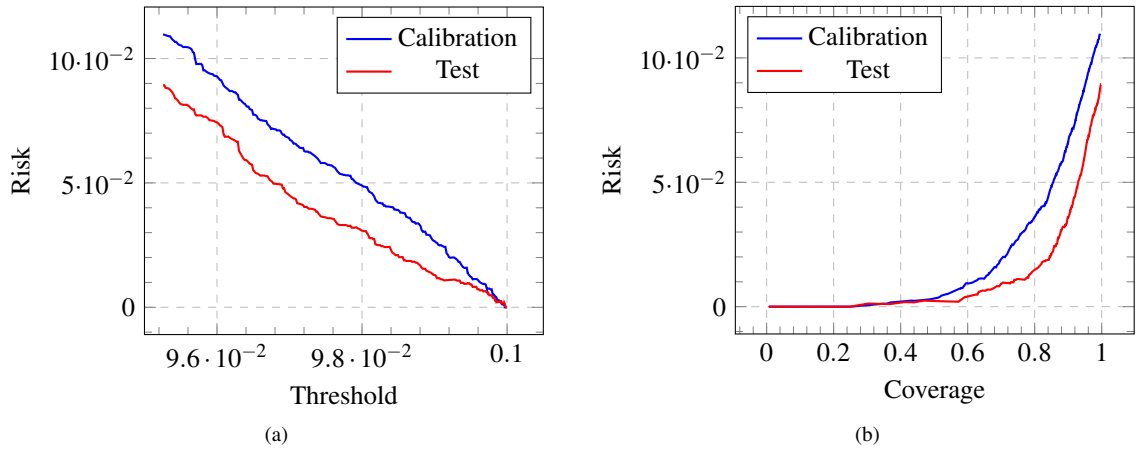


Figure V.3: (a) Assurance Evaluator risk curve, (b) Assurance Evaluator RC curve

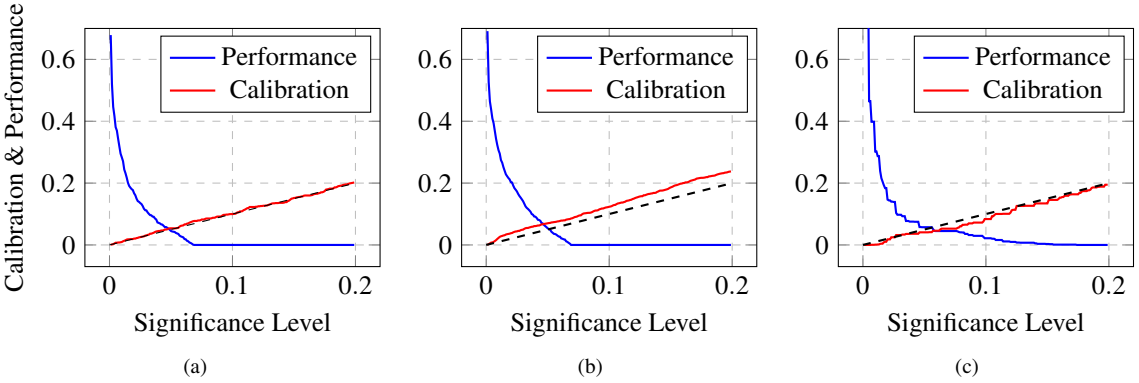


Figure V.4: (a) Baseline ICP on IID data, (b) Baseline ICP on sequential data, (c) Combination of p-values based on the min ECDF

V.5.4 Validity

ICP is proven to be valid when the input data are IID and exchangeable regardless the choice of the significance level and NCM. The first problem we work on is to recover the validity in cases where the input data are dependent. We examine the property, $P(y_{l+1} \notin \Gamma^{\varepsilon}) < \varepsilon$ in the baseline ICP using the test set containing IID and the test set containing sequences. For this, we plot the performance and calibration curves shown in Figure V.4. For different values of the significance level, the calibration curve show the percentage of test data with their ground truth class not contained in their prediction set, while the performance curve shows the percentage of test data that lead to prediction sets of more than one class. ICP is valid in the case of IID data but it under-estimates the true error-rate when data are sequential.

We evaluate the validity of ICP using the Expected Calibration Error (ECE). A well-calibrated ICP computes prediction sets with significance levels that are representative of the true error-rate. Formally a model

is well-calibrated when

$$\mathbb{P}(y_{l+1} \in \Gamma^{\varepsilon} | 1-\varepsilon = p) = p, \qquad \forall p \in [0,1] \tag{V.22}$$

where p is the actual prediction accuracy. However, ε is a continuous random variable so the probability in (V.22) cannot be approximated using finitely many samples. According to (V.22) a measure of miscalibration can be expressed as $\mathop{\mathbb{E}}_{\varepsilon}[|\mathbb{P}(y_{l+1} \in \Gamma^{\varepsilon} | 1-\varepsilon = p) - p|]$. The *Expected Calibration Error* (ECE) [11] computes an approximation of this expected value across samples of the significance level:

$$\text{ECE} = \frac{1}{M} \sum_{i=1}^{M} |\text{acc}(\varepsilon_i) + \varepsilon_i - 1| \tag{V.23}$$

Assuming n test examples, $\text{acc}(\varepsilon_i) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(y_i \in \Gamma_i^{\varepsilon})$ and ε_i the significance level samples. In this evaluation $\varepsilon_i = \frac{i}{1000} : i = 1, \ldots, 200$, as $\varepsilon > 0.2$ would not have any practical use in most CPS applications. Table V.3 shows the calibration results of ICP when we combine multiple sequential inputs in a sliding windows of different size. For comparison when the baseline ICP is used, ECE is 0.042.

Table V.3: ECE Comparison

		Sliding Window Size							
Method		2	3	4	5	6	7	8	9
Merging	Arith Avg	0.018	0.009	0.006	0.006	0.007	0.009	0.010	0.011
	Geom Avg	0.038	0.035	0.032	0.029	0.027	0.025	0.023	0.022
	Min	0.090	0.122	0.147	0.166	0.182	0.195	0.206	0.216
	Max	0.014	0.033	0.043	0.050	0.055	0.059	0.062	0.065
CDF	Fisher	0.086	0.120	0.144	0.162	0.176	0.188	0.198	0.208
	Stouffer	0.126	0.180	0.214	0.235	0.252	0.264	0.275	0.285
	Stouffer W	0.111	0.157	0.188	0.208	0.223	0.235	0.244	0.252
	Min	0.018	0.012	0.007	0.008	0.009	0.011	0.014	0.016
	Cauchi	0.052	0.059	0.063	0.065	0.067	0.068	0.070	0.071
ECDF	Sum	0.026	0.029	0.027	0.026	0.025	0.024	0.023	0.023
	Product	0.023	0.025	0.025	0.025	0.025	0.025	0.025	0.024
	Min	0.018	0.014	0.012	0.010	0.009	0.008	0.009	0.010
	Max	0.026	0.030	0.029	0.027	0.024	0.021	0.018	0.015
Baseline ICP		0.042							

Combining p-values using the ECDF based methods consistently improve the calibration over the baseline approach and have among the lowest ECE of all methods we tried. The same is observed for the merging functions that confirms the literature remarks about their validity under arbitrary dependence regardless their simplicity. Larger sliding window sizes affect the combination functions in different ways and do not guarantee better calibration. For example a low p-value that corresponds to a correct class will remain in the history for a longer time and depending on the combination function can significantly lower the aggregate p-value. Combining p-values using the quantile combination approaches, with the exception of the order

statistics function min, produces prediction sets with large calibration error confirming their inability to deal with dependence between the performed statistical tests. Combining multiple p-values by using only their minimum value and transforming it into a p-value using the incomplete beta function seems to not be affected by the dependence structure of the inputs. However, when using the calibration sequences to capture these dependencies and learn the ECDF instead of using the incomplete beta function, further improves the calibration in sliding window sizes greater than six.

V.5.5 Selective Classification on Sequences

The assurance evaluator identifies when a prediction is trustworthy. ICP computes the credibility and confidence from the p-values of all classes [Eqs. (V.1), (V.2)] and the assurance evaluator combines them to minimize the risk for any given coverage. We compare the decision performance of the baseline ICP and ICP based on combining p-values from multiple inputs. We also investigate how the sliding window size affects the decision quality. This comparison is based on the AURC which evaluates the average risk for different coverage values. In this part of the evaluation we combine p-values using the ECDF-based approaches as they showed stability against dependence between subsequent inputs. Table V.4 shows the AURC value for all the ECDF methods and for different sliding windows. For comparison the computed AURC for the baseline ICP is 0.011.

Table V.4: AURC Results

	ECDF			
Sliding Window Size	Sum	Product	Min	Max
2	0.007	0.007	0.007	0.007
3	0.005	0.005	0.005	0.006
4	0.004	0.004	0.004	0.005
5	0.004	0.004	0.004	0.004
6	0.003	0.003	0.004	0.004
7	0.003	0.003	0.004	0.004
8	0.003	0.003	0.003	0.004
9	0.003	0.003	0.003	0.003
Baseline	0.011			

All four alternatives show that when predictions are based on sliding windows of more than one input, the average risk is always lower than in the case of predictions based on a single input at a time. Moreover, the size of the sliding window also affects the risk. Our evaluation results show that predictions based on larger sliding windows have lower average risk. Figure V.5 shows the RC curves based on the four ECDF methods with sliding window size 9 compared with the RC curve produced with the baseline ICP.

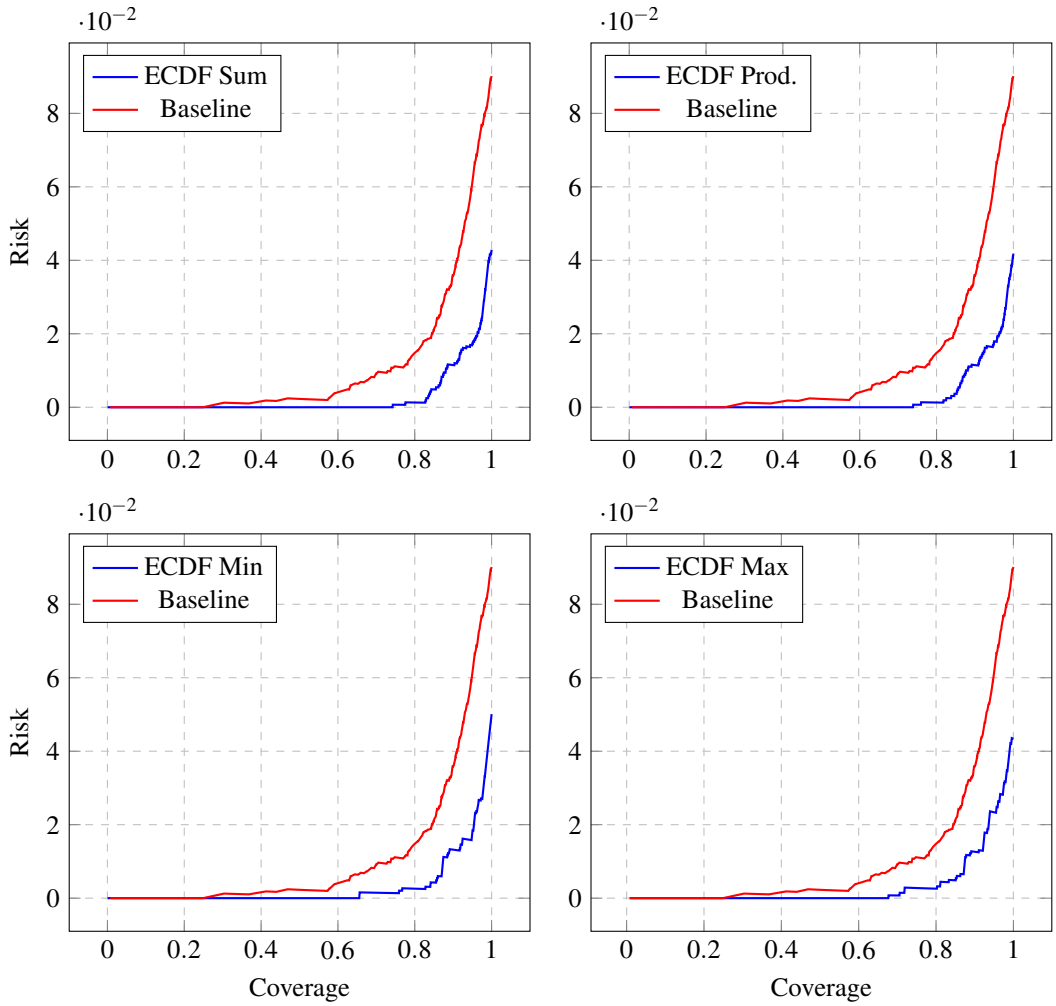


Figure V.5: Risk-Coverage Curves

V.6 Concluding Remarks

CPS use machine learning components for dynamic tasks that are hard to model such as perception of the environment. These components make predictions with a non-zero error-rate which makes their use in safety critical systems challenging. We designed a selective classifier that evaluates the trustworthiness of each prediction based on credibility and confidence values computed by ICP. In order to make efficient use of ICP we used a siamese network to map high-dimensional inputs to appropriate embedding representations. To recover the validity of ICP when subsequent inputs are time-correlated we combined the computed p-values using different multiple hypothesis testing methods. The experimental results using the GTSRB dataset, first demonstrate that taking into account one input at a time lead to over-confident classifiers. When p-values from more than one input data are combined using either merging functions or quantile functions based on ECDFs, we can recover the validity of the prediction sets. The approach is optimized to minimize the risk

given a data coverage. The evaluation results showed that the use of more than one subsequent inputs is beneficial and larger sliding window sizes lead to lower risk. The use of more than one subsequent inputs is also beneficial for computing the credibility and confidence needed for the selective classifer. Classifications based on multiple inputs experience less risk for the same coverage compared to classifications based on single inputs. Comparison between different window sizes show that larger sliding windows lead to lower risk.

CHAPTER VI

Reliable Probability Intervals for Classification

VI.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 [4].

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. 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 [21]. 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 probability intervals 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 [21, 33, 201–204]. In this chapter 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 the previous chapters we measured the likelihood of a prediction based on p-values. 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. As shown in Chapter III, 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 difference to true probabilities is in the inequality in the above definition that introduces the notion of bounded error-rate in the ICP framework. In this chapter the problem is the same as in chapter III, to compute a well-calibrated confidence metric, but using probability intervals that provide better intuition.

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 [205, 206], 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 [12] and temperature scaling [4] proposes ways of post-processing the outputs of already trained models to produce calibrated confidence measures. In [14, 15], 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 [21–23]. 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 [207], 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 that we compute 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. Then, we implement four different taxonomies that split the low-dimensional data into categories based on their similarity. Our third contribution is the implementation of 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 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 keeping into account the low-latency and low-power properties to meet the resource constraints of the variety of use cases [208].

VI.2 Problem

A perception component in an autonomous system aims to observe and interpret the environment in order to provide information for decision-making. For example, a DNN can be used for classifying traffic signs in autonomous vehicles. The problem is to complement the prediction of the DNN 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 Venn Prediction (VP) framework can produce predictions with well-calibrated confidence intervals that guarantee to include the true probabilities for each class output to occur [21]. 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 [21, 202, 204]. The additional problem we are considering is the computation of appropriate embedding representations that can lead to more efficient VPs. The main idea is to use distance metric learning and enable DNNs to learn a lower-dimensional representation for each input on an embedding space where the Euclidean distance between the input representations is a measure of similarity between the original inputs themselves. Using such representations we define taxonomies to form categories of similar input data. 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 we assign pseudo-labels to the unlabeled data that can be classified confidently. During execution time, input data arrive one by one. Their conformity with the training set is evaluated using the ICP framework. Using hypothesis testing, the labels that are less likely to describe the newly arrived data are rejected and the rest are used to update the class distribution in the categories.

VI.3 Probability Intervals based on Distance Metric Learning

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 [21, 23]. 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 [21]). 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 [202, 203], 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 [89, 166].

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. IVP assumes that all the examples $z_1, \ldots, z_{l+1}$ are independent and identically distributed (IID) generated from the same but usually unknown probability distribution. The available training examples are split into two parts: the *proper training set* with q examples and the calibration set with $l-q$ examples. 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 [21] the probability intervals assigned to each classification by the VP are well-calibrated regardless of the choice of the Venn taxonomy and this holds in practice for IVP as well [202]. 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 [209] we proposed four different Venn taxonomies based on distance metric learning. 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, $r_{l+1} = \text{Net}(x_{l+1})$ 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

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

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:

$$k_{l+1} = \hat{y}_{l+1} \times (k - \left\lfloor \frac{k}{c} \right\rfloor) + |i \in \Omega : i \neq \hat{y}_{l+1}| \tag{VI.2}$$

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 $k - \lfloor \frac{k}{c} \rfloor$ 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:

$$k_{l+1} = \arg\min_{j=1,\ldots,c} d(r_{l+1}, \mu_j) \tag{VI.3}$$

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:

$$k_{l+1} = 2 \times \arg\min_{j=1,\ldots,c} d(r_{l+1}, \mu_j) + h, \tag{VI.4}$$

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

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:

$$p(Y_j) = \frac{|\{(x^*, y^*) \in k_{l+1} : y^* = Y_j\}|}{|k_{l+1}|}. \tag{VI.5}$$

k_{l+1} will always be non-empty as it will contain at least the new example x_{l+1}. This creates a probability distrubution 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:

$$j_{best} = \arg\max_{j=1,\ldots,c} \overline{p(Y_j)} \tag{VI.6}$$

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, $P_{l+1}^{k_i} = \{p^{k_i}(Y_j) : k_i \in \{k_1, \ldots, k_n\}, Y_j \in \{Y_1, \ldots, Y_c\}\}$. 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 Figure VI.1.

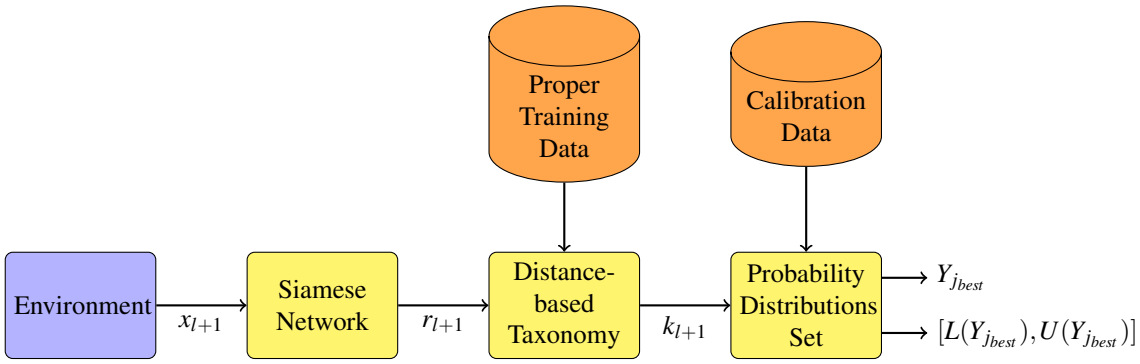


Figure VI.1: IVP classifier based on distance metric learning

Algorithm 5 – 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. VI.1, VI.2, VI.3, VI.4
Train f using $(x, y) \in (X, Y)$ ▷ Training
// Compute the representations
$V = f(X)$
$V^c = f(X^c)$
for each (v_i^c, y_i^c) in $(V^c, Y^c), i = 1..l-m$ **do**
    Compute the assigned category k_i using the chosen taxonomy ▷ Calibration
    Add y_i^c to k_i
end for
Store the resulted class distribution of each category

Algorithm 6 – Assurance Monitoring and Classification with Static Categories.

Require: Taxonomy from Eqs. VI.1, VI.2, VI.3, VI.4
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: $L(Y_j) = \dfrac{|\{(x^*, y^*) \in k_{l+1} : y^* = Y_j\}|}{|k_{l+1}| + 1}$
4: $U(Y_j) = \dfrac{|\{(x^*, y^*) \in k_{l+1} : y^* = Y_j\}| + 1}{|k_{l+1}| + 1}$
5: **end for**
6: Classify x_{l+1} into $j_{best} = \arg\max_{j=1,\ldots,c} \overline{p(Y_j)}$
7: Return probability interval $[L(Y_{j_{best}}), U(Y_{j_{best}})]$

VI.4 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 $P(y_{l+1} \notin \Gamma^{\varepsilon}) < \varepsilon$, 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 [189].

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 Figure VI.2. 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 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}.

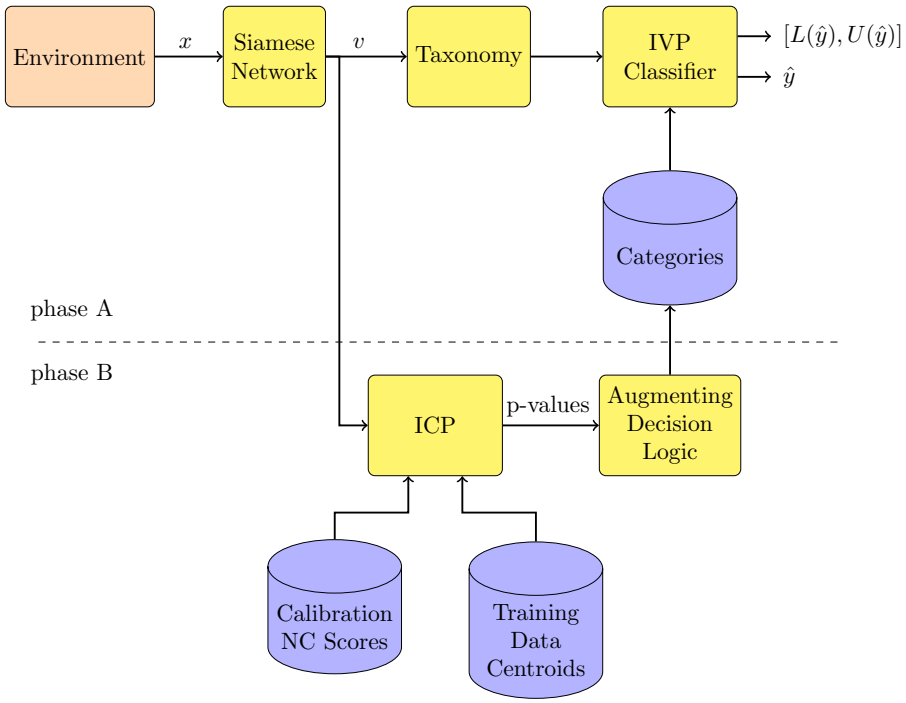


Figure VI.2: Execution time workflow

The usage of ICP is similar like in the previous chapters and it is based on distance metric learning. The proper training set and the calibration set used for ICP are the same that were used to train the siamese network and assign data to categories for the IVP framework. The same siamese network that was trained for IVP is used to define the ICP NCMs. In Chapter III, I presented different NCMs based on low-dimensional embedding representation. According to the application a particular NCM may be more suitable. The advantage of the nearest centroid NCM, in Equation III.4 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 Equation III.2. 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:

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

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:

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

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

Algorithm 7 – Assurance Monitoring and Classification with Dynamic Categories.

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

VI.5 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

$$E_n = \sum_{i=1}^{n} err_i, \tag{VI.7}$$

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

- cumulative lower and upper error probabilities

$$LEP_n = \sum_{i=1}^{n} [1 - U(\hat{y}_i)], \qquad UEP_n = \sum_{i=1}^{n} [1 - L(\hat{y}_i)] \tag{VI.8}$$

To compare the IVP implementations 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 (VI.6). 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

$$accuracy = 1 - \frac{E_n}{n}. \tag{VI.9}$$

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) [210]. NLL is the simplest out of the two and only considers the probability assigned to the predicted class in (VI.6). It is computed as

$$NLL = -\sum_{i=1}^{n} \sum_{j=1}^{c} t_i^j \log(o_i^j), \tag{VI.10}$$

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

$$BS = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} (o_i^j - t_i^j)^2 \tag{VI.11}$$

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. [211] 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, [212] 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. We evaluate the size of the probability intervals by computing the average interval diameter as

$$D = \frac{\sum_{i=1}^{n} U(\hat{y}_i) - \sum_{i=1}^{n} L(\hat{y}_i)}{n} \tag{VI.12}$$

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

$$\mathbb{P}(\hat{y} = Y | \hat{p} = p) = p, \qquad \forall p \in [0,1] \tag{VI.13}$$

However, $\hat{p}$ is a continuous random variable so the probability in (VI.13) cannot be approximated using finitely many samples. According to (VI.13) a measure of miscalibration can be expressed as $\mathbb{E}_{\hat{p}}[|\mathbb{P}(\hat{y} = y|\hat{p} = p) - p|]$. The *Expected Calibration Error* (ECE) [11] computes an approximation of this expected value across bins:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)| \tag{VI.14}$$

where $|B_m|$ is the number of samples in bin B_m, n is the total number of samples and $\text{acc}(B_m)$ and $\text{conf}(B_m)$ are the accuracy and confidence of bin B_m respectively as defined in [11]. 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) [11] and is computed as:

$$\text{MCE} = \max_{m \in \{1,\ldots,M\}} |\text{acc}(B_m) - \text{conf}(B_m)| \tag{VI.15}$$

VI.6 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.

VI.6.1 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 [170]. 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 [213]. 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 [214], 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 [215]. 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 [214], 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 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.

VI.6.2 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 [204]. 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.

VI.6.3 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 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 [39]. A commonly used metric of the separation between class clusters is the *silhouette coefficient* [175]. 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 in Table VI.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.

Table VI.1: Silhouette Coefficient Comparison

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

For illustration, the cumulative upper and lower error probabilities as well as the cumulative error are

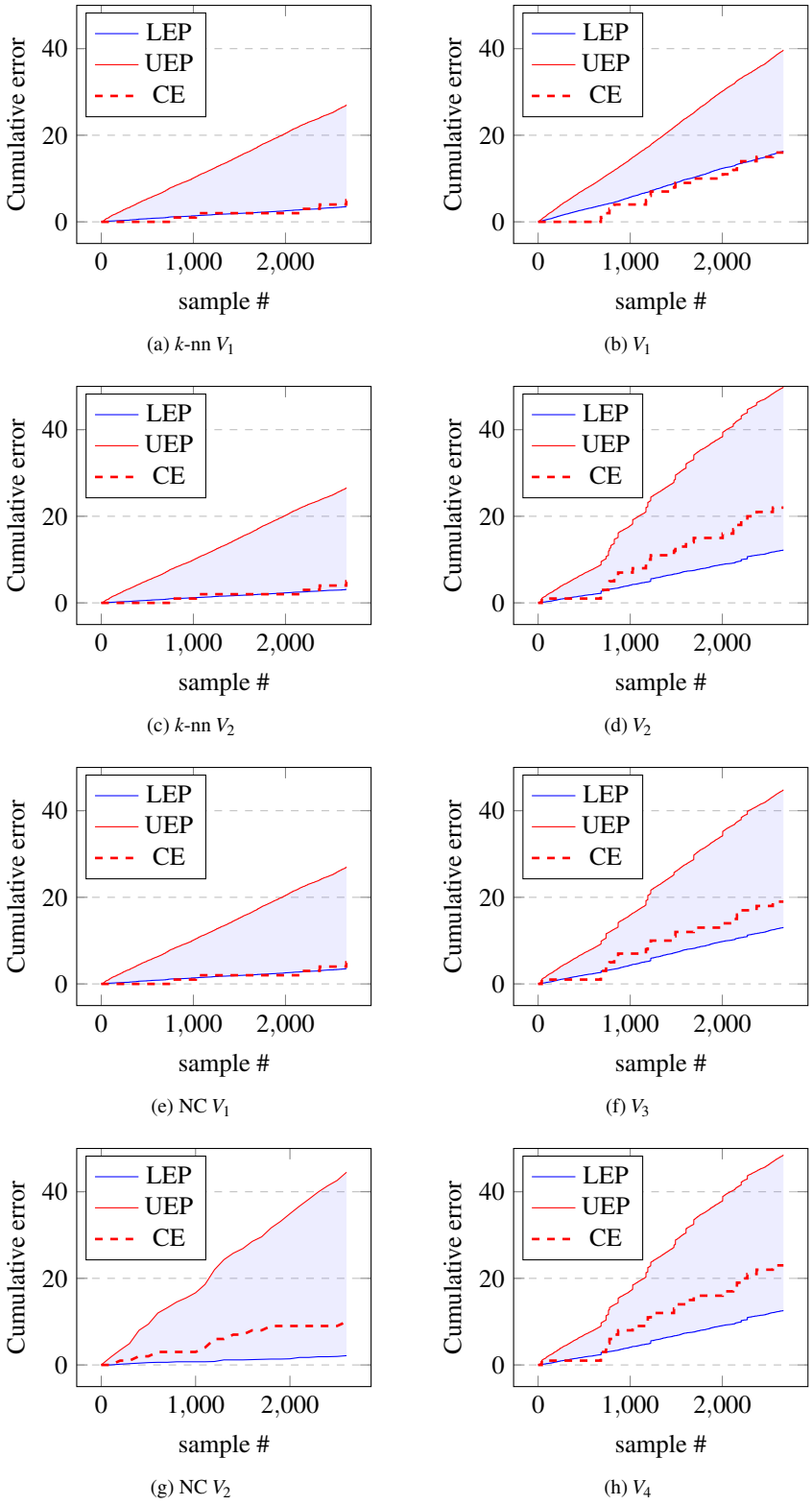
plotted on the same axis in Figure VI.3 for all the taxonomies used in our comparison and test data from the GTSRB dataset.

Table VI.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

The evaluation results are shown in Table VI.2. For both 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.

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, because the use of $k-d$ trees [176] for fast k-NN computation.



(a) k-nn V_1 (b) V_1

(c) k-nn V_2 (d) V_2

(e) NC V_1 (f) V_3

(g) NC V_2 (h) V_4

Figure VI.3: Cumulative error intervals comparison between our taxonomies and the literature baselines on the GTSRB dataset.

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 [176] 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.

VI.6.4 IVP with Dynamic Categories

IVP implementations typically form the categories during design time using labeled data and the categories remain the same during execution. In Section VI.4, 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 VI.5. 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 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 VI.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 Figure VI.4. The plots show the cumulative lower 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 [21]. Both the static and

Table VI.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

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, get extended. This not only lead to probability intervals that remain valid, but they get narrower over time compared to their static counterparts.

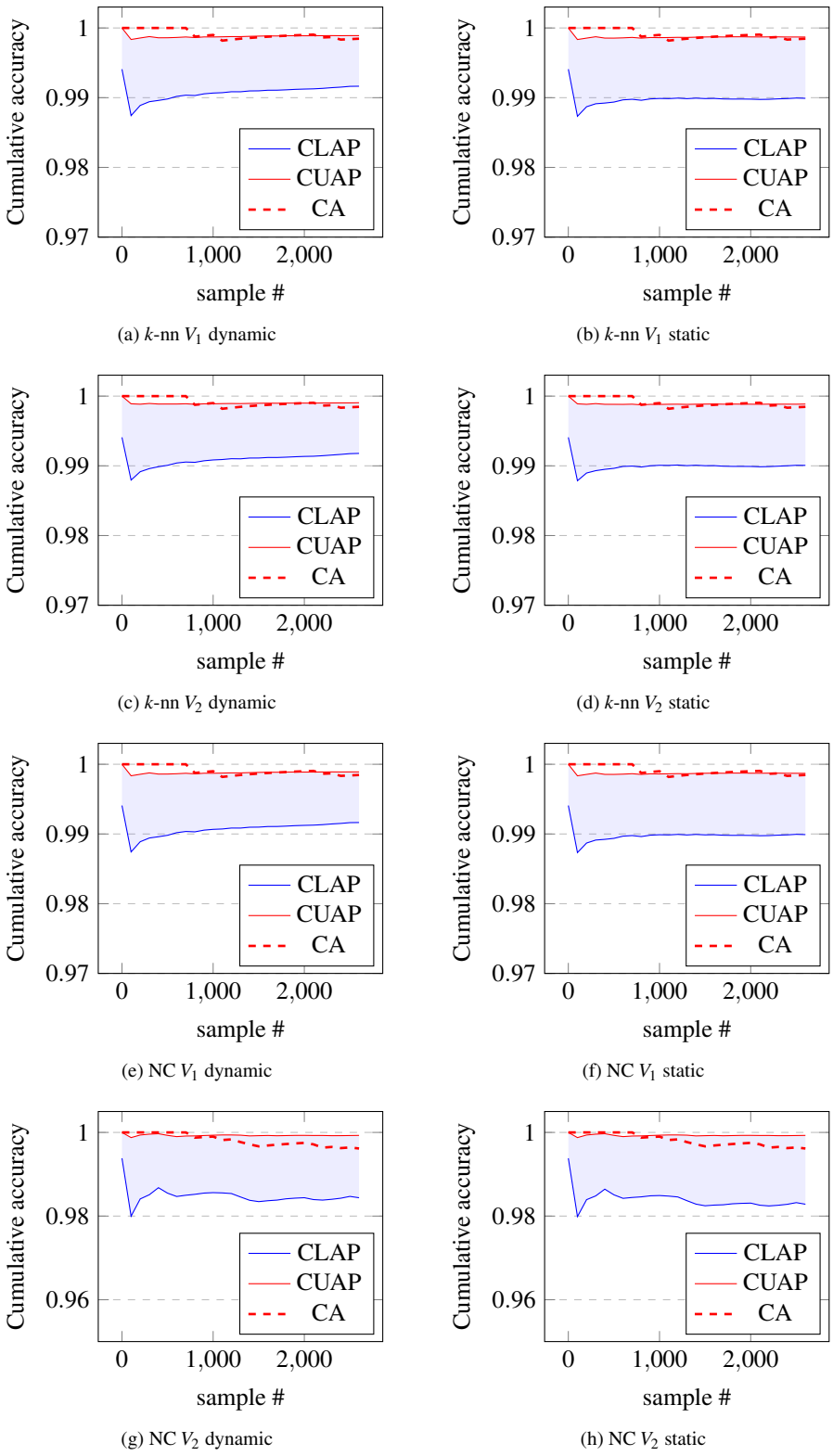
The results for both CPS related datasets are shown in Table VI.3. Each of the metrics used for evaluation is explained in Section VI.5. As we noted earlier the nearest centroid-based taxonomies are not a good choice for the botnet attack detection dataset as the clusters produced by the siamese network do not have clear centroids so in this particular application we focus on the results produced by the k-NN based taxonomies that have a 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 Equation VI.6. 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 and 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. That is 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.

VI.7 Concluding Remarks

Although DNNs offer advanced capabilities, they must be complemented by engineering methods and practices for them to provide accurate measures of prediction confidence. For classification tasks, the IVP framework computes probability intervals that contain the probability of the prediction's correctness by examining the underlying model's accuracy on similar data. We presented computationally efficient algorithms based on appropriate embedding representations learned by siamese networks 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 well-calibrated. 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.



(a) k-nn V_1 dynamic

(b) k-nn V_1 static

(c) k-nn V_2 dynamic

(d) k-nn V_2 static

(e) NC V_1 dynamic

(f) NC V_1 static

(g) NC V_2 dynamic

(h) NC V_2 static

Figure VI.4: Cumulative probability intervals comparison between the dynamic and static taxonomies on the GTSRB dataset.

CHAPTER VII

Conclusions

The increased use of autonomous systems in dynamic and complex environments has created many research problems in the CPS domain. Modern LEC architecture have the capacity to achieve high accuracy in fully autonomous systems. However in most cases they lack transparency and a way to reason for their decisions. Due to this, there needs to be a larger emphasis on techniques regarding prediction confidence calibration and explainability. For them to be used in real life applications they need to be complemented with a mechanism that can evaluate the risk of each autonomous decision in a way that humans can comprehend. This dissertation proposes different methods for the computation of well-calibrated assurance metrics based on the Inductive Conformal Prediction and Inductive Venn Predictors framework for CPS that utilize LECs. The methods we developed aim to solve the problems introduced by LEC in safety-critical systems by utilizing algorithms that are proven to produce well-calibrated predictions as well as distance metric learning methods for this to be practical in the majority of autonomous applications that are high-dimensional. Additionally incorporating techniques that express the prediction confidence as true probabilities helps humans interpret decision uncertainties and better trust the autonomous actions. Furtermore, the effectiveness of our methods in areas like calibration, efficiency and accuracy is supported by extensive evaluation on multiple real-life datasets. Even though extensions of the presented methods are still needed to achieve robustness in more complicated applications, we hope this work will help future contributions in this research area.

CHAPTER VIII

List of Publications

Published

1. **D. Boursinos** and X. Koutsoukos. Assurance Monitoring of Cyber-Physical Systems with Machine Learning Components. In *Proceedings of TMCE 2020*, Dublin, Ireland , May 2020.

2. **D. Boursinos** and X. Koutsoukos. Trusted Confidence Bounds for Learning Enabled Cyber-Physical Systems. In *Workshop for Assured Autonomous Systems 2020*, San Francisco, CA, May 2020. (Best Paper Award)

3. **D. Boursinos** and X. Koutsoukos. Improving Prediction Confidence in Learning-Enabled Autonomous Systems. In *DDDAS 2020*, Boston, MA , October 2020.

4. **D. Boursinos** and X. Koutsoukos. Assurance Monitoring of Learning Enabled Cyber-Physical Systems Using Inductive Conformal Prediction based on Distance Learning. In *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, May 2021.

5. D. Stojcsics, **D. Boursinos**, N. Mahadevan, X. Koutsoukos, G. Karsai. Fault-Adaptive Autonomy in Systems with Learning-Enabled Components. In *Sensors*, August 2021.

6. **D. Boursinos** and 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)*, pp. 1–7, August 2021.

7. **D. Boursinos** and X. Koutsoukos. Selective Classification of Sequential Data Using Inductive Conformal Prediction. In *2022 IEEE International Conference on Assured Autonomy (ICAA)*, Puerto Rico, March 2022.

In Preparation

1. **D. Boursinos** and X. Koutsoukos. Inductive Venn Predictors based on Distance Learning with Dynamic Categories for Reliable Probability Intervals.

BIBLIOGRAPHY

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[2] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[3] R. Baheti and H. Gill, "Cyber-physical systems," *The impact of control technology*, vol. 12, no. 1, pp. 161–166, 2011.

[4] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1321–1330, JMLR.org, 2017.

[5] R. H. Rasshofer, M. Spies, and H. Spies, "Influences of weather phenomena on automotive laser radar systems," *Advances in Radio Science: ARS*, vol. 9, p. 49, 2011.

[6] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in neural information processing systems*, pp. 6402–6413, 2017.

[7] M. Hein, M. Andriushchenko, and J. Bitterwolf, "Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 41–50, 2019.

[8] A. H. Murphy, "A new vector partition of the probability score," *Journal of applied Meteorology*, vol. 12, no. 4, pp. 595–600, 1973.

[9] M. H. DeGroot and S. E. Fienberg, "The comparison and evaluation of forecasters," *Journal of the Royal Statistical Society: Series D (The Statistician)*, vol. 32, no. 1-2, pp. 12–22, 1983.

[10] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *Proceedings of the 22nd international conference on Machine learning*, pp. 625–632, 2005.

[11] M. P. Naeini, G. F. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *Proceedings of the... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, vol. 2015, p. 2901, NIH Public Access, 2015.

[12] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pp. 61–74, MIT Press, 1999.

[13] J. Zhang and Y. Yang, "Probabilistic score estimation with piecewise logistic regression," in *Proceedings of the twenty-first international conference on Machine learning*, p. 115, 2004.

[14] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," in *Advances in Neural Information Processing Systems*, pp. 13991–14002, 2019.

[15] A. Kumar, P. S. Liang, and T. Ma, "Verified uncertainty calibration," in *Advances in Neural Information Processing Systems*, pp. 3792–3803, 2019.

[16] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers," in *Icml*, vol. 1, pp. 609–616, Citeseer, 2001.

[17] B. Zadrozny and C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, (New York, NY, USA), pp. 694–699, ACM, 2002.

[18] M. P. Naeini and G. F. Cooper, "Binary classifier calibration using an ensemble of piecewise linear regression models," *Knowledge and information systems*, vol. 54, no. 1, pp. 151–170, 2018.

[19] X. Jiang, M. Osl, J. Kim, and L. Ohno-Machado, "Calibrating predictive model estimates to support personalized medicine," *Journal of the American Medical Informatics Association*, vol. 19, no. 2, pp. 263–274, 2012.

[20] M. Sun and S. Cho, "Obtaining calibrated probability using roc binning," *Pattern Analysis and Applications*, vol. 21, no. 2, pp. 307–322, 2018.

[21] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic Learning in a Random World*. Berlin, Heidelberg: Springer-Verlag, 2005.

[22] G. Shafer and V. Vovk, "A tutorial on conformal prediction," *J. Mach. Learn. Res.*, vol. 9, pp. 371–421, June 2008.

[23] V. Balasubramanian, S.-S. Ho, and V. Vovk, *Conformal Prediction for Reliable Machine Learning: Theory, Adaptations and Applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1st ed., 2014.

[24] H. Papadopoulos, V. Vovk, and A. Gammermam, "Conformal prediction with neural networks," in *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, vol. 2, pp. 388–395, IEEE, 2007.

[25] N. Papernot and P. McDaniel, "Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning," 2018.

[26] U. Johansson, H. Boström, and T. Löfström, "Conformal prediction using decision trees," in *2013 IEEE 13th international conference on data mining*, pp. 330–339, IEEE, 2013.

[27] S. Bhattacharyya, "Confidence in predictions from random tree ensembles," *Knowledge and Information Systems*, vol. 35, pp. 391–410, 5 2013.

[28] D. Devetyarov and I. Nouretdinov, "Prediction with confidence based on a random forest classifier," in *Artificial Intelligence Applications and Innovations*, (Berlin, Heidelberg), pp. 37–44, Springer Berlin Heidelberg, 2010.

[29] L. Makili, J. Vega, S. Dormido-Canto, I. Pastor, and A. Murari, "Computationally efficient svm multi-class image recognition with confidence measures," *Fusion Engineering and Design*, vol. 86, no. 6, pp. 1213 – 1216, 2011. Proceedings of the 26th Symposium of Fusion Technology (SOFT-26).

[30] D. Boursinos and X. Koutsoukos, "Improving prediction confidence in learning-enabled autonomous systems," in *InfoSymbiotics/DDDAS2020*, 2020.

[31] H. Papadopoulos, V. Vovk, and A. Gammerman, "Regression conformal prediction with nearest neighbours," *Journal of Artificial Intelligence Research*, vol. 40, pp. 815–840, 2011.

[32] U. Johansson, H. Boström, T. Löfström, and H. Linusson, "Regression conformal prediction with random forests," *Machine Learning*, vol. 97, no. 1-2, pp. 155–176, 2014.

[33] H. Papadopoulos and H. Haralambous, "Reliable prediction intervals with regression neural networks," *Neural Networks*, vol. 24, no. 8, pp. 842–851, 2011.

[34] J. Sun and C. R. Loader, "Simultaneous confidence bands for linear regression and smoothing," *The Annals of Statistics*, vol. 22, no. 3, pp. 1328–1345, 1994.

[35] J. Goldsmith, S. Greven, and C. Crainiceanu, "Corrected confidence bands for functional data using principal components," *Biometrics*, vol. 69, no. 1, pp. 41–51, 2013.

[36] U. Johansson, H. Linusson, T. Löfström, and H. Boström, "Model-agnostic nonconformity functions for conformal classification," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2072–2079, 5 2017.

[37] D. Boursinos and X. Koutsoukos, "Assurance monitoring of cyber-physical systems with machine learning components," in *Tools and Methods of Competitive Engineering*, pp. 27–38, 2020.

[38] S. Bhattacharyya, "Confidence in predictions from random tree ensembles," in *2011 IEEE 11th International Conference on Data Mining*, pp. 71–80, IEEE, 2011.

[39] G. E. Hinton, "Learning multiple layers of representation," *Trends in cognitive sciences*, vol. 11, no. 10, pp. 428–434, 2007.

[40] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt, "Practical and optimal lsh for angular distance," in *Advances in neural information processing systems*, pp. 1225–1233, 2015.

[41] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, (San Francisco, CA, USA), pp. 518–529, Morgan Kaufmann Publishers Inc., 1999.

[42] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *2006 47th annual IEEE symposium on foundations of computer science (FOCS'06)*, pp. 459–468, IEEE, 2006.

[43] D. Devetyarov and I. Nouretdinov, "Prediction with confidence based on a random forest classifier," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pp. 37–44, Springer, 2010.

[44] G. W. Brier, "Verification of Forecasts Expressed in Terms of Probability," *Monthly Weather Review*, vol. 78, pp. 1–3, 01 1950.

[45] V. N. Balasubramanian, R. Gouripeddi, S. Panchanathan, J. Vermillion, A. Bhaskaran, and R. Siegel, "Support vector machine based conformal predictors for risk of complications following a coronary drug eluting stent procedure," in *2009 36th Annual Computers in Cardiology Conference (CinC)*, pp. 5–8, IEEE, 2009.

[46] P. Toccaceli, I. Nouretdinov, and A. Gammerman, "Conformal predictors for compound activity prediction," in *Symposium on Conformal and Probabilistic Prediction with Applications*, pp. 51–66, Springer, 2016.

[47] H. Yu, J. Yang, and J. Han, "Classifying large data sets using svms with hierarchical clusters," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 306–315, 2003.

[48] L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, "Large scale kernel machines. neural information processing series," 2007.

[49] K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, H. Cui, and E. Y. Chang, "Parallelizing support vector machines on distributed computers," in *Advances in Neural Information Processing Systems*, pp. 257–264, 2008.

[50] K. Woodsend and J. Gondzio, "Hybrid mpi/openmp parallel linear support vector machine training," *The Journal of Machine Learning Research*, vol. 10, pp. 1937–1953, 2009.

[51] Y. You, H. Fu, S. L. Song, A. Randles, D. Kerbyson, A. Marquez, G. Yang, and A. Hoisie, "Scaling support vector machines on modern hpc platforms," *Journal of Parallel and Distributed Computing*, vol. 76, pp. 16–31, 2015.

[52] H.-F. Yu, C.-J. Hsieh, K.-W. Chang, and C.-J. Lin, "Large linear classification when data cannot fit in memory," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 5, no. 4, pp. 1–23, 2012.

[53] A. Z. Zeyuan, C. Weizhu, W. Gang, Z. Chenguang, and C. Zheng, "P-packsvm: Parallel primal gradient descent kernel svm," in *2009 Ninth IEEE International Conference on Data Mining*, pp. 677–686, IEEE, 2009.

[54] H. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik, "Parallel support vector machines: The cascade svm," *Advances in neural information processing systems*, vol. 17, pp. 521–528, 2004.

[55] K. Veropoulos, C. Campbell, N. Cristianini, *et al.*, "Controlling the sensitivity of support vector machines," in *Proceedings of the international joint conference on AI*, vol. 55, p. 60, 1999.

[56] R. Akbani, S. Kwek, and N. Japkowicz, "Applying support vector machines to imbalanced datasets," in *European conference on machine learning*, pp. 39–50, Springer, 2004.

[57] G. Wu and E. Y. Chang, "Adaptive feature-space conformal transformation for imbalanced-data learning," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 816–823, 2003.

[58] G. Wu and E. Y. Chang, "Class-boundary alignment for imbalanced dataset learning," in *ICML 2003 workshop on learning from imbalanced data sets II, Washington, DC*, pp. 49–56, 2003.

[59] T. Gartner, *Kernels for structured data*, vol. 72. World Scientific, 2008.

[60] F. Provost and P. Domingos, "Tree induction for probability-based ranking," *Machine learning*, vol. 52, no. 3, pp. 199–215, 2003.

[61] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, Oct 2001.

[62] H. Papadopoulos, "Inductive conformal prediction: Theory and application to neural networks," in *Tools in artificial intelligence*, Citeseer, 2008.

[63] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *Advances in neural information processing systems*, pp. 521–528, 2003.

[64] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM review*, vol. 38, no. 1, pp. 49–95, 1996.

[65] R. T. Rockafellar, *Convex Analysis*. Princeton university press, 1970.

[66] M. Schultz and T. Joachims, "Learning a distance metric from relative comparisons," in *Advances in neural information processing systems*, pp. 41–48, 2004.

[67] C. Cortes and V. Vapnik, "Support vector machine," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[68] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in neural information processing systems*, pp. 1473–1480, 2006.

[69] S. Parameswaran and K. Q. Weinberger, "Large margin multi-task metric learning," in *Advances in neural information processing systems*, pp. 1867–1875, 2010.

[70] D. Kedem, S. Tyree, F. Sha, G. R. Lanckriet, and K. Q. Weinberger, "Non-linear metric learning," in *Advances in neural information processing systems*, pp. 2573–2581, 2012.

[71] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification.," *Journal of Machine Learning Research*, vol. 10, no. 2, 2009.

[72] K. Q. Weinberger and L. K. Saul, "Fast solvers and efficient implementations for distance metric learning," in *Proceedings of the 25th international conference on Machine learning*, pp. 1160–1167, 2008.

[73] L. Torresani and K.-c. Lee, "Large margin component analysis," in *Advances in neural information processing systems*, pp. 1385–1392, 2007.

[74] N. Nguyen and Y. Guo, "Metric learning: A support vector approach," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 125–136, Springer, 2008.

[75] K. Park, C. Shen, Z. Hao, and J. Kim, "Efficiently learning a distance metric for large margin nearest neighbor classification," in *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pp. 453–458, 2011.

[76] M. Der and L. K. Saul, "Latent coincidence analysis: A hidden variable model for distance metric learning," in *Advances in Neural Information Processing Systems*, pp. 3230–3238, 2012.

[77] A. Globerson and S. T. Roweis, "Metric learning by collapsing classes," in *Advances in neural information processing systems*, pp. 451–458, 2006.

[78] S. Shalev-Shwartz, Y. Singer, and A. Y. Ng, "Online and batch learning of pseudo-metrics," in *Proceedings of the twenty-first international conference on Machine learning*, p. 94, 2004.

[79] Y. Bengio, *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[80] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: An astounding baseline for recognition," in *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 512–519, 2014.

[81] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, Springer, 2014.

[82] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[83] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[84] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1701–1708, 2014.

[85] R. Salakhutdinov and G. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure," in *Artificial Intelligence and Statistics*, pp. 412–419, 2007.

[86] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in neural information processing systems*, pp. 737–744, 1994.

[87] Y. LeCun *et al.*, "Generalization and network design strategies," *Connectionism in perspective*, vol. 19, pp. 143–155, 1989.

[88] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 539–546, IEEE, 2005.

[89] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 1735–1742, IEEE, 2006.

[90] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*, pp. 84–92, Springer, 2015.

[91] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

[92] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *European conference on computer vision*, pp. 499–515, Springer, 2016.

[93] X. He, Y. Zhou, Z. Zhou, S. Bai, and X. Bai, "Triplet-center loss for multi-view 3d object retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1945–1954, 2018.

[94] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE transactions on information theory*, vol. 8, no. 2, pp. 179–187, 1962.

[95] L. G. Roberts, *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963.

[96] A. Guzmán, "Decomposition of a visual scene into three-dimensional bodies," in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pp. 291–304, 1968.

[97] D. Waltz, "Understanding line drawings of scenes with shadows," in *The psychology of computer vision*, Citeseer, 1975.

[98] M. B. Clowes, "On seeing things," *Artificial intelligence*, vol. 2, no. 1, pp. 79–116, 1971.

[99] A. Macworth, "Interpreting pictures of polyhedral scenes," *Artificial intelligence*, vol. 4, no. 2, pp. 121–137, 1973.

[100] S. Ullman, *Interpretation of Visual Motion*. MIT Press, 1979.

[101] Y. Lamdan and H. J. Wolfson, "Geometric hashing: A general and efficient model-based recognition scheme," in *[1988 Proceedings] Second International Conference on Computer Vision*, pp. 238–249, 1988.

[102] S. Ullman and R. Basri, "Recognition by linear combinations of models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 10, pp. 992–1006, 1991.

[103] D. Weinshall and C. Tomasi, "Linear and incremental acquisition of invariant shape models from image sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 5, pp. 512–517, 1995.

[104] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1150–1157, Ieee, 1999.

[105] J. S. Beis and D. G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," in *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, CVPR '97, (USA), p. 1000, IEEE Computer Society, 1997.

[106] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[107] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, vol. 1, pp. I–I, IEEE, 2001.

[108] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[109] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[110] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, pp. 886–893, IEEE, 2005.

[111] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *European conference on computer vision*, pp. 589–600, Springer, 2006.

[112] E. Osuna, R. Freund, and F. Girosit, "Training support vector machines: an application to face detection," in *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, pp. 130–136, IEEE, 1997.

[113] V. Vapnik, *The nature of statistical learning theory*. Springer, 1995.

[114] C. J. Burges, "Simplified support vector decision rules," in *ICML*, vol. 96, pp. 71–77, 1996.

[115] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 144–152, 1992.

[116] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[117] R. Xiao, L. Zhu, and H.-J. Zhang, "Boosting chain learning for object detection," in *Proceedings Ninth IEEE International Conference on Computer Vision*, pp. 709–715, IEEE, 2003.

[118] T. Kohonen, *Self-organization and associative memory*. Springer-Verlag, 1984.

[119] G. Burel and D. Carel, "Detection and localization of faces on digital images," *Pattern Recognition Letters*, vol. 15, no. 10, pp. 963 – 967, 1994.

[120] S.-H. Lin, S.-Y. Kung, and L.-J. Lin, "Face recognition/detection by probabilistic decision-based neural network," *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 114–132, 1997.

[121] R. Vaillant, C. Monrocq, and Y. Le Cun, "Original approach for the localisation of objects in images," *IEE Proceedings-Vision, Image and Signal Processing*, vol. 141, no. 4, pp. 245–250, 1994.

[122] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 20, no. 1, pp. 23–38, 1998.

[123] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[124] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), vol. 25, pp. 1097–1105, Curran Associates, Inc., 2012.

[125] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[126] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[127] I. Endres and D. Hoiem, "Category independent object proposals," in *European Conference on Computer Vision*, pp. 575–588, Springer, 2010.

[128] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2189–2202, 2012.

[129] J. Carreira and C. Sminchisescu, "Cpmc: Automatic object segmentation using constrained parametric min-cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1312–1328, 2011.

[130] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[131] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, pp. 1458–1465, IEEE, 2005.

[132] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 2169–2178, IEEE, 2006.

[133] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[134] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[135] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[136] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[137] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[138] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[139] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 4, pp. 640–651, 2017.

[140] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

[141] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

[142] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7310–7311, 2017.

[143] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[144] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, "Light-head r-cnn: In defense of two-stage object detector," *arXiv preprint arXiv:1711.07264*, 2017.

[145] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

[146] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[147] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[148] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.

[149] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[150] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[151] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[152] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "CSPNet: A new backbone that can enhance learning capability of cnn," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 390–391, 2020.

[153] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation.," in *AAAI*, pp. 13001–13008, 2020.

[154] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[155] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[156] S. Liu, D. Huang, *et al.*, "Receptive field block net for accurate and fast object detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 385–400, 2018.

[157] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

[158] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "CBAM: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

[159] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[160] H. Law and J. Deng, "CornerNet: Detecting objects as paired keypoints," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 734–750, 2018.

[161] A. Newell, Z. Huang, and J. Deng, "Associative embedding: End-to-end learning for joint detection and grouping," in *Advances in neural information processing systems*, pp. 2277–2287, 2017.

[162] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European conference on computer vision*, pp. 483–499, Springer, 2016.

[163] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "CenterNet: Keypoint triplets for object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6569–6578, 2019.

[164] D. Boursinos and X. Koutsoukos, "Trusted confidence bounds for learning enabled cyber-physical systems," in *Workshop on Assured Autonomous Systems at SP2020*, 2020.

[165] D. Boursinos and 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*, vol. 35, no. 2, p. 251–264, 2021.

[166] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2, Lille, 2015.

[167] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*, pp. 84–92, Springer, 2015.

[168] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[169] E. Kiplagat, "American rhetoric (online speech bank)." https://americanrhetoric.com/speechbank.htm.

[170] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, vol. 32, pp. 323 – 332, 2012. Selected Papers from IJCNN 2011.

[171] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[172] I. Melekhov, J. Kannala, and E. Rahtu, "Siamese network features for image matching," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 378–383, IEEE, 2016.

[173] H. Xuan, A. Stylianou, and R. Pless, "Improved embeddings with easy positive triplet mining," *arXiv preprint arXiv:1904.04370*, 2019.

[174] C. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, pp. 10–21, 1 1949.

[175] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53 – 65, 1987.

[176] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, pp. 509–517, Sept. 1975.

[177] F. Darema, "Grid computing and beyond: The context of dynamic data driven applications systems," *Proceedings of the IEEE*, vol. 93, no. 3, pp. 692–697, 2005.

[178] A. Aved and E. Blasch, "Multi-int query language for dddas designs," *Procedia Computer Science*, vol. 51, pp. 2518–2532, 12 2015.

[179] B. Uzkent, M. J. Hoffman, and A. Vodacek, "Integrating hyperspectral likelihoods in a multidimensional assignment algorithm for aerial vehicle tracking," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 9, pp. 4325–4333, 2016.

[180] E. Blasch, G. Seetharaman, and F. Darema, "Dynamic Data Driven Applications Systems (DDDAS) modeling for automatic target recognition," in *Automatic Target Recognition XXIII* (F. A. Sadjadi and A. Mahalanobis, eds.), vol. 8744, pp. 165 – 174, International Society for Optics and Photonics, SPIE, 2013.

[181] D. Allaire, J. Chambers, R. Cowlagi, D. Kordonowy, M. Lecerf, L. Mainini, F. Ulker, and K. Willcox, "An offline/online dddas capability for self-aware aerospace vehicles," *Procedia Computer Science*, vol. 18, pp. 1959 – 1968, 2013. 2013 International Conference on Computational Science.

[182] R. M. Fujimoto, N. Celik, H. Damgacioglu, M. Hunter, D. Jin, Y.-J. Son, and J. Xu, "Dynamic data driven application systems for smart cities and urban infrastructures," in *2016 Winter Simulation Conference (WSC)*, pp. 1143–1157, IEEE, 2016.

[183] D. Boursinos and X. Koutsoukos, "Improving prediction confidence in learning-enabled autonomous systems," in *Dynamic Data Driven Applications Systems* (F. Darema, E. Blasch, S. Ravela, and A. Aved, eds.), (Cham), pp. 217–224, Springer International Publishing, 2020.

[184] M. Kääriäinen and J. Langford, "A comparison of tight generalization error bounds," in *Proceedings of the 22nd International Conference on Machine Learning*, pp. 409–416, Association for Computing Machinery, 2005.

[185] D. Boursinos and X. Koutsoukos, "Selective classification of sequential data using inductive conformal prediction," in *2022 IEEE International Conference on Assured Autonomy (ICAA) (ICAA'22)*, (virtual, Puerto Rico), Mar. 2022.

[186] R. El-Yaniv *et al.*, "On the foundations of noise-free selective classification.," *Journal of Machine Learning Research*, vol. 11, no. 5, 2010.

[187] Y. Wiener and R. El-Yaniv, "Agnostic selective classification," *Advances in neural information processing systems*, vol. 24, pp. 1665–1673, 2011.

[188] Y. Geifman, G. Uziel, and R. El-Yaniv, "Bias-reduced uncertainty estimation for deep neural classifiers," in *7th International Conference on Learning Representations, ICLR, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[189] G. Shafer and V. Vovk, "A tutorial on conformal prediction.," *Journal of Machine Learning Research*, vol. 9, no. 3, 2008.

[190] V. Vovk and R. Wang, "Combining p-values via averaging," *Biometrika*, vol. 107, no. 4, pp. 791–808, 2020.

[191] B. Rüger, "Das maximale signifikanzniveau des tests: "lehneho ab, wennk untern gegebenen tests zur ablehnung führen"," *Metrika*, vol. 25, pp. 171–178, 1978.

[192] L. Rüschendorf, "Random variables with maximum sums," *Advances in Applied Probability*, vol. 14, no. 3, pp. 623–632, 1982.

[193] P. Toccaceli, "Conformal predictor combination using Neyman–Pearson Lemma," in *Conformal and Probabilistic Prediction and Applications*, pp. 66–88, PMLR, 2019. ISSN: 2640-3498.

[194] P. Toccaceli and A. Gammerman, "Combination of conformal predictors for classification," in *Proceedings of the Sixth Workshop on Conformal and Probabilistic Prediction and Applications* (A. Gammerman, V. Vovk, Z. Luo, and H. Papadopoulos, eds.), vol. 60 of *Proceedings of Machine Learning Research*, pp. 39–61, PMLR, 13–16 Jun 2017.

[195] R. A. Fisher *et al.*, "224a: Answer to question 14 on combining independent tests of significance.," 1948.

[196] S. A. Stouffer, E. A. Suchman, L. C. Devinney, S. A. Star, and R. M. Williams Jr., *The American soldier: Adjustment during army life. (Studies in social psychology in World War II), Vol. 1*. Princeton Univ. Press, 1949.

[197] T. Lipták, "On the combination of independent tests," *Magyar Tud Akad Mat Kutato Int Kozl*, vol. 3, pp. 171–197, 1958.

[198] S. M. Ross, *Introduction to Probability Models*. Academic Press, 2014.

[199] Y. Liu and J. Xie, "Cauchy combination test: A powerful test with analytic p-value calculation under arbitrary dependency structures," *Journal of the American Statistical Association*, vol. 115, no. 529, pp. 393–402, 2020. PMID: 33012899.

[200] K. Aslansefat, S. Kabir, A. Abdullatif, V. Vasudevan, and Y. Papadopoulos, "Toward improving confidence in autonomous vehicle software: A study on traffic sign recognition systems," *Computer*, vol. 54, no. 8, pp. 66–76, 2021.

[201] V. Vovk, G. Shafer, and I. Nouretdinov, "Self-calibrating probability forecasting.," in *NIPS*, pp. 1133–1140, 2003.

[202] A. Lambrou, I. Nouretdinov, and H. Papadopoulos, "Inductive venn prediction," *Annals of Mathematics and Artificial Intelligence*, vol. 74, no. 1-2, pp. 181–201, 2015.

[203] A. Lambrou, H. Papadopoulos, I. Nouretdinov, and A. Gammerman, "Reliable probability estimates based on support vector machines for large multiclass datasets," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pp. 182–191, Springer, 2012.

[204] H. Papadopoulos, "Reliable probabilistic classification with neural networks," *Neurocomputing*, vol. 107, pp. 59–68, 2013.

[205] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," *arXiv preprint arXiv:1701.06548*, 2017.

[206] A. Kumar, S. Sarawagi, and U. Jain, "Trainable calibration measures for neural networks from kernel mean embeddings," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, (Stockholmsmässan, Stockholm Sweden), pp. 2805–2814, PMLR, 10–15 Jul 2018.

[207] H. Papadopoulos, V. Vovk, and A. Gammerman, "Regression conformal prediction with nearest neighbours," *J. Artif. Int. Res.*, vol. 40, p. 815–840, Jan. 2011.

[208] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[209] D. Boursinos and 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)*, pp. 1–7, 2021.

[210] G. W. Brier, "Verification of forecasts expressed in terms of probability," *Monthly Weather Review*, vol. 78, no. 1, pp. 1 – 3, 01 Jan. 1950.

[211] R. Benedetti, "Scoring rules for forecast verification," *Monthly Weather Review*, vol. 138, no. 1, pp. 203–211, 2010.

[212] R. Selten, "Axiomatic characterization of the quadratic scoring rule," *Experimental Economics*, vol. 1, no. 1, pp. 43–61, 1998.

[213] H. Mureşan and M. Oltean, "Fruit recognition from images using deep learning," *arXiv preprint arXiv:1712.00580*, 2017.

[214] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[215] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and Other Botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.