Trusted Confidence Bounds for Learning Enabled Cyber-Physical Systems

Dimitrios Boursinos
Institute for Software Integrated Systems
Vanderbilt University
Nashville TN, USA
dimitrios.boursinos@vanderbilt.edu

Xenofon Koutsoukos
Institute for Software Integrated Systems
Vanderbilt University
Nashville TN, USA
xenofon.koutsoukos@vanderbilt.edu

***Abstract*—Cyber-physical systems (CPS) can benefit by the use of learning enabled components (LECs) such as deep neural networks (DNNs) for perception and decision making tasks. However, DNNs are typically non-transparent making reasoning about their predictions very difficult, and hence their application to safety-critical systems is very challenging. LECs could be integrated easier into CPS if their predictions could be complemented with a confidence measure that quantifies how much we trust their output. The paper presents an approach for computing confidence bounds based on Inductive Conformal Prediction (ICP). We train a Triplet Network architecture to learn representations of the input data that can be used to estimate the similarity between test examples and examples in the training data set. Then, these representations are used to estimate the confidence of set predictions from a classifier that is based on the neural network architecture used in the triplet. The approach is evaluated using a robotic navigation benchmark and the results show that we can computed trusted confidence bounds efficiently in real-time.**

***Index Terms*—Cyber-physical systems, deep neural networks, assurance monitoring, conformal prediction, triplet, robot navigation**

I. Introduction

Machine learning components are being used by many cyber-physical system (CPS) applications because of their ability to handle dynamic and uncertain environments. Deep neural networks (DNNs), for example, are used for perception and decision making tasks in autonomous vehicles. 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 the system. 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.

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. Several approaches have been proposed for confidence estimation. Neural networks for classification, in particular, typically provide probability-like outputs using a softmax layer. However, these probabilities are typically overconfident even for inputs coming from the same distribution as the training data [5]. The reason is that the softmax probabilities are not well-calibrated meaning they do not provide a good estimate of the error rates. Several methods are proposed to compute well-calibrated confidence values. A class of methods aims to estimate scaling factors from the training data and use them to scale the softmax probabilities in order to compute well-calibrated confidence values and include temperature scaling [5], Platt scaling [10], and isotonic regression [15]. Although such methods can compute well-calibrated confidence values, using them in CPS requires selecting an appropriate confidence bound which ensures a very small error rate and at the same time limiting the number of inputs for which a confidence prediction cannot be made.

The approach presented in this paper is based on conformal prediction (CP) [1]. For classification, CP associates reliable confidence values with set predictions that may include multiple labels. The confidence measures are well-calibrated and can be computed in an online setting which is suitable for CPS applications. The online application of the approach is based on the inductive conformal prediction (ICP) for computational efficiency [8]. ICP leverages a calibration data set that is used to compute the confidence values of new previously unseen inputs efficiently. The confidence values rely on nonconformity functions computed using techniques such as k-Nearest Neighbors and Kernel Density Estimation. However, such approaches do not scale for high-dimensional inputs.

In our previous work, we used the ICP framework for assurance monitoring of CPS with machine learning components [3]. In order to handle high-dimensional inputs in real-time, the approach computes the nonconformity scores using the embedding representations produced by trained DNN models in lower dimensional spaces than the input space. The main contribution of this paper is a significant improvement in computing confidence bounds by employing a triplet network architecture to learn representations of the input data that can be used to estimate the similarity between test examples and examples in the training data set. Then, these representations are used to estimate the confidence of set predictions from a classifier that is based on the neural network architecture used in the triplet. In order to achieve the desired confidence, the LEC may need to generate more than one possible prediction. Sets with multiple predictions can be useful especially when the alternatives are provided to a human operator. However, in this paper, we focus on autonomous CPS and compute the optimal confidence bounds required for the LEC to generate single predictions.

Another contribution of the paper is the evaluation of the approach using the SCITOS-G5 robotic navigation benchmark. Our results show that we can compute trusted confidence bounds efficiently in real-time. We also compare the proposed approach with the approach presented in [3] which relies on applying ICP using representations learned by the model used to generate the predictions. The comparison shows for a chosen confidence bound, ICP using triplet produces fewer sets with multiple candidate labels, or equivalently there is a higher confidence bound that eliminate all the sets with multiple predictions.

The paper is organized a follows. Section II describes the problem. Section III reviews the Triplet Network architecture and the associated training algorithms. Section IV presents the methods for ICP using the Triplet. We evaluate the performance of the proposed approach in Section V and we discuss the conclusions in Section VI.

II. Problem

Learning-enabled components in CPS are used for taking decisions based on the state of the system and the environment. For a mobile robot, for example, an LEC can be used to navigate through a room while avoiding collisions with obstacles. A commonly used approach involves learning a model using training data and using the learned model for operation. We expect the system to perform better in scenarios similar to those used during the training phase. The problem we consider is the computation of a significance level along with each decision by the LEC. The significance level must be well-calibrated which means that it must be consistent with the expected error and ideally the expected error rate must be bounded. Moreover, CPS applications require minimizing the number of incorrect decisions while limiting the number of false alarms to enable efficient monitoring. Another requirement for CPS is that such monitoring must be performed in real-time often with limited computional resources.

III. Triplet Network

The ICP framework requires a way to estimate the similarity between the training data and a test input. The main idea in our approach is that we can do this efficiently by learning representations of the inputs for which the Euclidean distance is a suitable measure of similarity. The triplet network is a Deep Neural Network (DNN) architecture that is trained to compute appropriate representations for metric or distance learning [6].

A Triplet Network is composed using three copies of the same neural network with shared parameters as shown in Fig. 1. 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^+ are of the same class while x^- is of a different class. The last layer of the neural network computes a representation $Net(x)$. The objective is to maximize the distance between inputs of different classes $|Net(x) - Net(x^-)|$ and minimize the distance of inputs belonging to the same class $|Net(x) - Net(x^+)|$. To achieve this, training uses the loss function:

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

where α is the margin between positive and negative pairs.

The triplet network can be trained by randomly sampling anchor data from the training set and augmenting them with one training sample with the same label as the anchor's and one sample with a different label, randomly chosen. However, this method leads to slow training and low performance as samples that result to $|Net(x) - Net(x^-)| >> |Net(x) - Net(x^+)| + \alpha$ do not provide useful information. The training can be improved by carefully mining the training data [14]. 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 all the hard negative samples, meaning the samples that satisfy $|Net(x) - Net(x^-)| < |Net(x) - Net(x^+)|$.

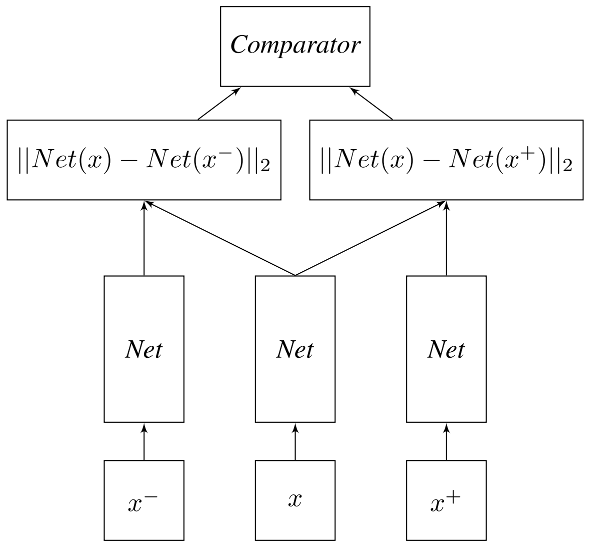

Figure 1: Triplet network structure [6]

A trained triplet network maps the raw data inputs to a lower dimensional space called *embedding space*. The distance between the representations $Net(x)$ of the inputs is a useful measure of similarity and can be used for classification by training a k-NN classifier on the embedding space of the training data. This distance can also be used by the ICP framework as described in the next section.

IV. Triplet-Based ICP

In this section, we briefly explain the Inductive Conformal Prediction (ICP) approach based on the Triplet Network

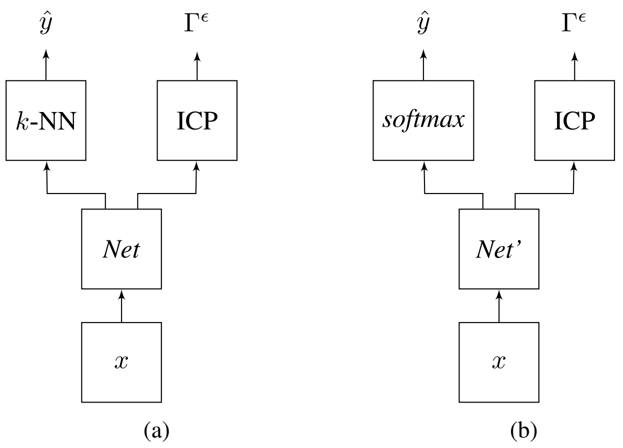

Figure 2: (a) LEC based on Triplet and (b) LEC based on DNN classifier

architecture. We consider a sequence of 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. ICP 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.

For a chosen significance level $\epsilon \in [0,1]$, the objective is to generate a set of possible labels Γ^ϵ for the input x_{l+1}, such that the probability of the correct label $y_{l+1} \notin \Gamma^\epsilon$ does not exceed ϵ. ICP is based on a *nonconformity measure* (NCM) which is a dissimilarity metric 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 [3], [1], [13], [12], [7]. The proposed approach uses the Triplet Network to estimate similarities by encoding the inputs to an embedding space where the Euclidean distance between two samples is a direct measure of similarity. NCMs defined in the embedding space include (1) the *k-Nearest Neighbors* (k-NN) [9], (2) the *one Nearest Neighbor* (1-NN) [13] and (3) the *Nearest Centroid* [1].

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 encodings $v_i = f(x_i)$ for the training data x_i. Given a test example 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 nonconformity of the test example x with a candidate label y is defined as:

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

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

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

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

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

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 NCM $a(x,y)$ is a measure of dissimilarity between a test input x with candidate label y and the training data $z_1 \ldots z_l$ as larger values would indicate higher "dissimilarity". However, this measure does not provide useful information by itself but it can be used by comparing it with NCM values 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 Triplet Network and the calibration set $(z_{m+1} \ldots z_l)$ of size $l - m$. The method first computes the NCMs $a(x_i, y_i)$, $i = m+1 \ldots l$ for the examples in the calibration set. Given a test example x with an unknown label y, the method forms a set $|\Gamma^\epsilon|$ of possible labels $\tilde{y}$ so that $P(y \notin |\Gamma^\epsilon|) < \epsilon$. For all the candidate labels $\tilde{y}$, ICP is based on the empirical p-value defined as

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

that computes the fraction of nonconformity scores of the calibration data that are equal or larger than the nonconformity score of a test input. A candidate label is added to Γ^ϵ if $p_j(x) > \epsilon$.

Depending on the desired significance level ϵ, there may be more than one possible label. Although these multiple labels can provide useful information, we assume we do not have a way to process more than one possible label and we want to minimize these cases. The objective of the monitoring algorithm is after receiving each input to compute a valid prediction that ensures a predefined error rate (based on ϵ) and limits the number of input examples for which a confident prediction cannot be made. The output of the monitor is defined as:

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

Depending on the size of the set Γ^ϵ, the monitor outputs $out = 1$ to indicate there is a single prediction with bounded confidence by ϵ. If the set Γ^ϵ is either empty or have multiple labels, the monitor raises alarms to indicate that no decision can be made. However, we distinguish between multiple and no predictions, because they may lead to different action in the system. An empty Γ^ϵ may be an indication that the input is out-of-distribution while multiple possible labels indicates that the accuracy of the underlying model is lower than the chosen ϵ.

Algorithm 1 – Monitoring Algorithm.

```
Input: training data $(X, Y)$, calibration data $(X^c, Y^c)$
Input: trained neural network $f$ with $l$ layers
Input: Nonconformity function $\alpha$
Input: test input $z$
Input: significance level threshold $\epsilon$
 1: // Compute the nonconformity scores for the calibration data offline
 2: $A = \{\alpha(x, y) : (x, y) \in (X^c, Y^c)\}$          ▷ Calibration
 3: // Generate prediction sets for each test data online
 4: for each label $j \in 1..n$ do
 5:     Compute the nonconformity score $\alpha(z, j)$
 6:     $p_j(z) = \frac{|\{\alpha \in A : \alpha \geq \alpha(z,j)\}|}{|A|}$          ▷ empirical $p$-value
 7:     if $p_j(z) \geq \epsilon$ then
 8:         Add $j$ to the prediction set $\Gamma^\epsilon$
 9:     end if
10: end for
11: if $|\Gamma^\epsilon| = 0$ then
12:     return 0
13: else if $|\Gamma^\epsilon| = 1$ then
14:     return 1
15: else
16:     return Reject
17: end if
```

V. Evaluation

In this section, we evaluate how the triplet network improves the computation of significance levels using ICP. For the comparison, we use metrics that include how well inputs for different classes are clustered in the embedding represenation learned by the Triplet, the validity and efficiency of the produced prediction sets $|\Gamma^\epsilon|$, the execution time of the monitoring algorithm, and the required memory.

A. Experimental Setup

We apply the proposed method to the SCITOS-G5 wall following robot navigation dataset [4]. The robot has the task of navigating around the room counter-clockwise in close proximity to the walls. It is equipped with 24 ultrasound sensors that are sampled at a rate of 9 samples per second. The possible actions 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. 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.

The triplet network is formed using three identical DNNs with shared weights as shown in Fig. 1. Since the inputs in the SCITOS-G5 dataset come from 24 sensors, we treat them as vectors and use a fully connected neural network. The parameters of the network architecture are shown in Table I. For the baseline approach the DNN of Table I is trained using the dedicated training data. The output of the hidden layer FC4, without taking into account the ReLu activation, is used to produce the representations for the baseline ICP. The FC5 is used as the baseline DNN classifier. The Triplet is trained using three copies of the same base DNN with hidden layers FC1-FC4. After training, only one of the three copies is used to generate the embeddings on the output of FC4 without any activation.

Layer	Size	Activation
FC 1	128	ReLu
FC 2	128	ReLu
FC 3	128	ReLu
FC 4	16	ReLu
FC 5	4	Softmax

Table I: The DNN architecture

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.

B. Triplet Performance

We first look at how well the triplet clusters the data on a 16-dimensional space. For comparison, we use the embedding space produced by the FC4 hidden layer of the DNN classifier. A commonly used metric of the separation between classes is the *Silhouette* [11]. 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 Triplet with the representations learned using the FC4 hidden layer of the fully connected DNN, we compute the mean silhouette over the training data and the validation data separately. In Table II, we see that the representations learned by the Triplet form better-defined clusters.

	Training Silhouette	Validation Silhouette
Triplet Embeddings	0.52	0.46
DNN Embeddings	0.17	0.17

Table II: Clustering comparison using the silhouette coefficient

Another way to evaluate the performance of the triplet network is to combine it with a k-Nearest Neighbors classifier as shown in Figure 2a, and compute the classification accuracy. A test input is converted to an embedding representation using the triplet and then the k-NN classifier with k=15 is used to make a prediction. The baseline DNN uses a softmax activation (Figure 2b) for the classification. The Table III shows the accuracy of the triplet combined with a k-NN compared to the softmax classifier both using the same base neural network architecture shown in Table I. The triplet network results in a better performance than the DNN classifier with softmax.

	Training Accuracy	Testing Accuracy
Triplet + k-NN	95%	91.2%
DNN + Softmax	91.93%	88.49%

Table III: Comparison of the classification accuracy between a k-NN classifier on the triplet embeddings and a baseline DNN with softmax layer

C. ICP and Assurance Monitoring

The SCITOS-G5 robot is a safety-critical CPS for which the controller needs to take decisions with a well-calibrated and valid predefined significance level. As a baseline, we consider a NCM defined based on the penultimate layer of a DNN to generate a representations that allow real-time monitoring [3]. We evaluate the performance of the triplet when used with the nonconformity functions presented in IV. In particular, we compare the calibration and validity of the predicted candidate label sets as well as we compute the bound ϵ that eliminates the sets Γ^ϵ with $|\Gamma^\epsilon| > 1$, that is the sets with multiple predictions.

First, we verify that the error rates of the monitoring algorithm are bounded by the significance level ϵ. We compute the percentage of incorrect predictions and we plot the cumulative error for different values of ϵ. In Figure 3, we plot the cumulative error for three different values of ϵ for the SCITOS-G5 dataset using the Nearest Centroid nonconformity function. We see that the three different significance levels bound the cumulative error rate well. Similar behavior is observed using the other two nonconformity functions.

In order to see how well-calibrated the confidence bounds are as well as how many sets with multiple candidate predictions are generated when $\epsilon \in [0.001, 0.4]$, we plot the calibration and performance curves in Figure 4. The number

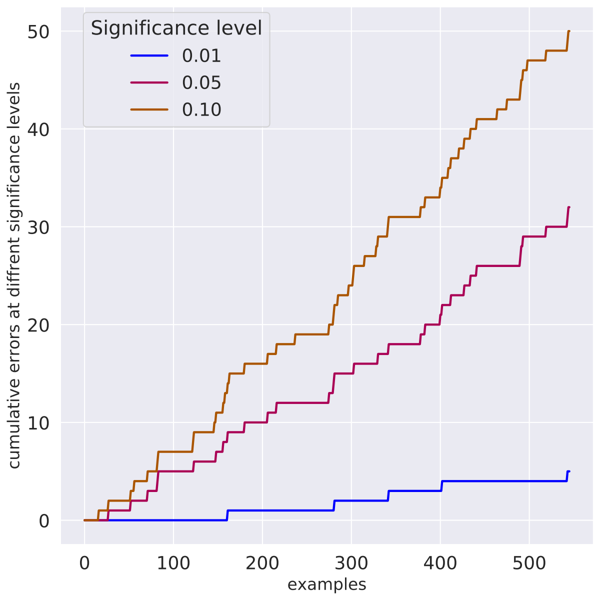

Figure 3: Cumulative error curve

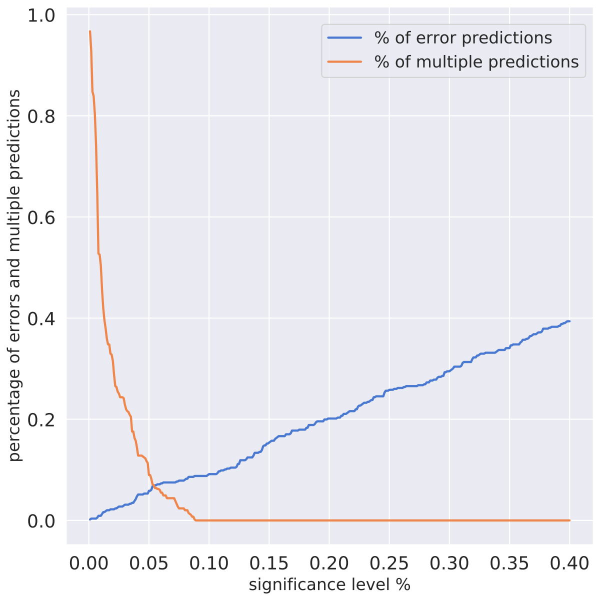

Figure 4: Calibration and performance curve

of multiple predictions decreases fast as ϵ increases. Further, the error rate is well-calibrated and increases linearly with ϵ.

Table IV reports the average execution time for each test input and the required memory using different nonconformity functions. The memory needed for the ICP application with each NC function is the memory required to store the DNN weights and the memory required to store the training data used by each NC function. In the k-NN and 1-NN case, the encodings of the training data are stored in a $k-d$ tree [2] that is used to compute efficiently the nearest neighbors. In the 1-NN case, it is required to find the nearest neighbor in the training data for each possible class which is

		Estimate ϵ		$\epsilon = 0.01$		$\epsilon = 0.05$		Runtime Requirements	
Architecture	NC Functions	ϵ	Errors	Errors	Multiples	Errors	Multiples	Memory	Time
Triplet	k-NN	0.089	9.3%	0%	100%	4%	11.9%	1.02 MB	1.4ms
	1-NN	0.087	6.8%	0.2%	44.3%	2.4%	13.2%	3.5 MB	2.8ms
	Nearest Centroid	0.092	8.8%	0.9%	50.4%	5.8%	9%	180 kB	1.1ms
FC DNN	k-NN	0.095	10.2%	0%	100%	1.5%	39.7%	1.02 MB	1.6ms
	1-NN	0.08	7.9%	1.4%	33.9%	2.5%	22.2%	3.5 MB	3ms
	Nearest Centroid	0.201	19.4%	1.6%	75.3%	1.6%	69%	180 kB	1.4ms

Table IV: Test results for different values of ϵ

computationally expensive resulting in larger execution time. The nearest centroid nonconformity function requires storing only the centroids for each class so the additional memory required is minimal. Both the triplet network architecture and the FC DNN we use as a baseline have the same network architecture and generate the same embedding size so the memory requirements are exactly the same between the two.

The evaluation results demonstrate that both the baseline FC DNN based ICP confidence monitor and the Triplet based ICP confidence monitor have well-calibrated error rates. The approach allows selecting the significance level to trade-off errors and alarms. When our objective is to minimize the number of alarms raised because of multiple candidate predictions, the Triplet with k-NN or Nearest Centroid NC functions satisfies it while keeping a higher significance level. When the significance level is selected, the Triplet based algorithm produces less sets of multiple candidate predictions. Finally, the memory requirements and the execution times for both approaches show that this monitoring system can be used for real-time CPS applications.

VI. Conclusion

DNN components are being used in Cyber-physical systems (CPS) to perform tasks like perception and control. However, their decision making process cannot be interpreted in a straightforward way and they cannot provide well-calibrated probabilities on the correctness of their predictions. This paper considers the problem of complementing the prediction of DNNs with a computation of confidence for real-time monitoring of CPS. We used the Inductive Conformal Prediction framework for classification tasks to produce sets of predictions with an error rate bounded by a chosen significance level. The evaluation results demonstrate that the addition of Triplet to the ICP framework produces well-calibrated probabilities as well as less prediction sets with more than one candidate decisions for a given significance level. The Triplet networks have been used on applications with higher dimensional input space such as images. Thus our future work is to scale our suggested assurance monitoring approach to be used on ICP applications that deal with images, like the camera outputs on self driving vehicles. Moreover, since there are many NC functions that can be used it is natural to see if we can combine the prediction sets produced by each of them on a test input using ensemble methods with voting to produce a narrower prediction set.

Acknowledgment

The material presented in this paper is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) through contract number FA8750-18-C-0089 and the Air Force Office of Scientific Research (AFOSR) DDDAS through contract number FA9550-18-1-0126. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or AFOSR

References

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

[2] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, Sept. 1975.

[3] D. Boursinos and X. Koutsoukos. Assurance monitoring of cyber-physical systems with machine learning components. *arXiv preprint arXiv:2001.05014*, 2020.

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

[5] 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, pages 1321–1330. JMLR.org, 2017.

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

[7] 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)*, pages 2072–2079, May 2017.

[8] H. Papadopoulos. Inductive conformal prediction: Theory and application to neural networks. In *Tools in artificial intelligence*. IntechOpen, 2008.

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

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

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

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

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

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

[15] 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, pages 694–699, New York, NY, USA, 2002. ACM.