ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20201112163731
Preliminary communication

Neural Network-Based Model for Classification of Faults During Operation of a Robotic Manipulator

Sandi BARESSI ŠEGOTA*, Nikola ANĐELIĆ, Zlatan CAR, Mario ŠERCER

Abstract: The importance of error detection is high, especially in modern manufacturing processes where assembly lines operate without direct supervision. Stopping the faulty operation in time can prevent damage to the assembly line. Public dataset is used, containing 15 classes, 2 types of faultless operation and 13 types of faults, with 463 force and torsion datapoints. Four different methods are used: Multilayer Perceptron (MLP) selected due to high classification performance, Support Vector Machines (SVM) commonly used for a low number of datapoints, Convolutional Neural Network (CNN) known for high performance in classification with matrix inputs and Siamese Neural Network (SNN) novel method with high performance in small datasets. Two classification tasks are performed-error detection and classification. Grid search is used for hyperparameter variation and F_1 score as a metric, with a 10 fold cross-validation. Authors propose a hybrid system consisting of SNN for detection and CNN for fault classification.

Keywords: Convolutional Neural Network; Multilayer Perceptron; Robot Fault Detection; Siamese Neural Network; Support Vector Machine

1 INTRODUCTION

The importance of fault detection within industrial robotic manipulators is extremely high. With the rise of remote and autonomously operating manufacturing lines the importance of the ability to timely detect possible faults in the operation of robotic manipulators is growing [1], as well as acting accordingly, either by stopping the operation of the manipulator or adjusting it. Artificial Intelligence (AI) applications are used to a great effect in many fields, including robotics [2-5]. Due to the quick classification capabilities of trained ML algorithms and the high accuracies that can be achieved with such methods, they could be used in the task at hand. Still, a problem arises with the need for large amounts of data needed to train such algorithms. With the operation of robotic manipulators, due to high costs of equipment and the need for almost constant faultless operation inside realistic manufacturing environments, collection of enough data points describing faulty operation can be hard to achieve [6]. Because of this, this research focuses on using a relatively small dataset, consisting of only 463 data points (when datasets in the usual training use thousands of data points), in an attempt to achieve good scores with ML methods. Furthermore, to lower the number of classes when detecting the type of fault the research tests the hybrid model in which one neural network is used to detect the normal and faulty operation, and another to detect the type of the fault.

Costa et al. [7] demonstrate failure detection in robotic arms through multiple methods such as statistical modeling, machine learning, and hybrid gradient boosting. The authors conclude that various models provide the best solutions in different situations. Pinto and Cerquitelli [8] (2019) discuss possible solutions in robotics oriented fault detection with the goal of predictive maintenance. Authors utilize algorithms such as survival analysis, extremely randomized trees, and k-nearest neighbors to achieve the pre-emptive prediction of errors. Furthermore, the authors have validated their process in a service-oriented solution. Cheng et al. [9] (2019) show the use of unsupervised learning in detecting failures in robots based on the analysis of current signals. These techniques are validated using experimental data collected from industrial robot systems on which they show high effectiveness and accuracy. Mitrevski et al. [10] (2019) propose a data-driven fault detection and classification system based on modified Boltzmann machines, each representing a distribution of sliding window correlations between a pair of correlated measurements, with the obtained models showing 0.886 precision score and 0.756 recall score. Piltan et al. [11] (2020) demonstrate an SVM-based neural adaptive variable structure observer with the goal of fault classification and fault-tolerant control of a robot manipulator. The effectiveness of the developed algorithm is validated using a PUMA robot manipulator, with the performance of the fault detection being approved, on average, by 27 to 29.2%.

Yu et al. [12] (2019) show the use of SNN for camera pose estimation and visual servoing, with sub-millimeter precision. The system has a 97.5% accuracy on the validation task of the VGA-connector insertion task, without any force sensing mechanism. Chang et al. [13] (2019) achieve high accuracy in visual object tracking using a convolutional SNN, comparing its performance to CNN and discriminative correlation filter (DFC) methods. Ogul et al. [14] (2019) rank artificial intelligence-based methods on robot-assisted surgery using kinematic sensors and propose the use of a modified Siamese network that compares kinematic samples and selects one with higher skill.

A lack of papers testing the possibility of using one-shot learning in fault detection is apparent, despite a high number of papers demonstrating the use of machine learning algorithms for such a goal. Furthermore, the comparisons of SNN to better-established machine learning classification algorithms, such as MLP, SVM, or CNN, have not been performed on the problem of robot fault detection in recent years.

The aim of this research is the development of a system that can not only recognize if the fault has happened based on the aforementioned data, but also classify the type of the error. It should be noted that the error detection must have an extremely high, if not perfect, score-both when calculated as precision and recall (or combined into the F_1

score explained in the Methodology section). This is because any misclassified faulty operation can be costly. In case of the false positive, where the normal operation is classified as faulty, the process of stopping the robotic manipulator operation, along with the possibility of the need for human intervention, can cause time-intensive delays in the manufacturing. This is especially relevant at such points in which crucial operations, on which other parts of the manufacturing chain depend, are performed by a low number of industrial manipulators - or even a single one, completely stopping the entire manufacturing process. Furthermore, false negative, in which a faulty operation is classified as the normal operation can be disastrous as the operation monitored by the AI system would not be stopped, and it would cause the operation to continue despite the possible obstructions or collisions. An instance of this case could cause damages to not only the manipulator, but also to other elements of the manufacturing line, and could cause the need for costly repairs and the stoppage of manufacture for extended periods. The algorithm for detecting the type of the error does not need to be held to such rigorous standards, and minor errors in classification can be tolerated as that part would serve to inform the separate element, most probably a human operator, of the type of the fault that has happened. The purpose of this is the planning of actions needed to remove the issues. In the case of an experienced worker, familiar with the plant, knowing the type of obstruction or collision could shorten the time needed for repairs. Types of faults detected can also be automatically logged and used to provide information for planning adjustments to floor plans in such manufacture cells in which the positioning of elements causes faults to happen often.

2 METHODOLOGY

In this section, the used dataset and algorithms are presented. The preparation of data for SNNs is also described.

2.1 Dataset Description

Dataset consists of 463 data points. These data points each consist of 15 force and torque measurements in three directions (x, y, and z), measured at the tip of the manipulator's end-effector [6]. These measurements are performed as a time series, with 315 ms between measurements. With 15 measurements this means that each data point consists of 4.725 seconds of measurements. Each data point is provided as a matrix:

$$M = \begin{pmatrix} F_x^1 & F_y^1 & F_z^1 & T_x^1 & T_y^1 & T_z^1 \\ F_x^2 & F_y^2 & F_z^2 & T_x^2 & T_y^2 & T_z^2 \\ \cdots & & & & & \\ F_x^{15} & F_y^{15} & F_z^{15} & T_x^{15} & T_y^{15} & T_z^{15} \end{pmatrix} \quad (1)$$

463 data points that exist inside the dataset can be divided into classes. The class distribution is shown in Fig. 1 [6].

On the y axis, various classes contained within the dataset are shown, along with their number. Both numbers of instances within original dataset classes and classes grouped for binary classification are shown.

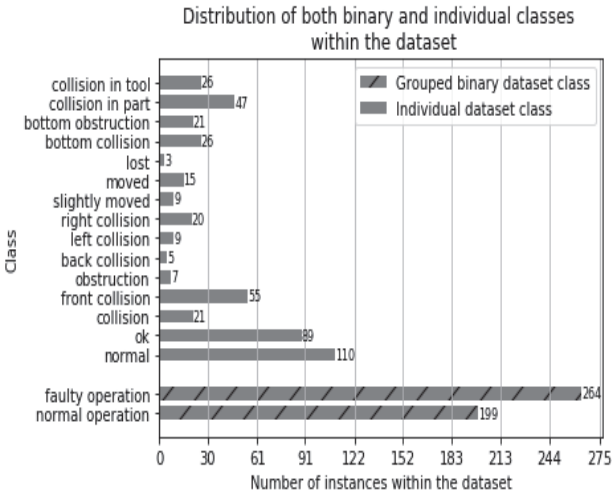

Figure 1 The distribution of data inside the dataset

Data has been collected by the dataset authors using a Selective Compliance Articulated Robot Arm (SCARA) type robot. The end effector of the robot consisted of a robotic grasper, with integrated sensors for data collection. Data collection was performed with various objects, in other words-various load sizes, to obtain a higher distribution of measured torque and force values of the end-effector.

2.2 Machine Learning Methodology

The following subsections will describe the methods used in this research. Some notes which are mutual to all of the methods will be presented here.

Each of the four methods used is a so-called supervised learning method. This signifies that the training process used is observed and the adjustments are actively made to the method parameters depending on the current error [15]. As such, all the methods are separated into the training and testing stages. During the training stage, the aforementioned adjustments are made to the internal method parameters to minimize the error. The testing step happens after the training is finished, on the obtained models, and its purpose is to determine how well the model fits the data.

The training and testing are performed on the two separate parts of the input dataset. The dataset is separated randomly, without repetitions, with 90% of the data being used for training and 10% of the data being used for testing. This is the same for each of the methods used, except for SNN which uses a different dataset, which was derived from the original.

Each of the methods has a set of given hyperparameters. These hyperparameters have an extremely large influence on the performance of the trained models [16]. As these hyperparameters differ between the methods, the description of them, as well as the values used are given within the appropriate subsections. Determination of all hyperparameters was performed using the grid search algorithm. The grid search algorithm works in such a way that it takes all the combinations of pre-

defined hyperparameter values and trains the models using each of the hyperparameter combinations [17].

Cross-Validation has been performed for all of the algorithms described in this section. The cross-validation algorithm used was K-FOLD cross-validation. K-FOLD cross-validation works by separating the set into K folds and then repeating the training K times. Each of these iterations uses one of the folds as a testing set, with the remaining $K-1$ folds being used as the training set. In this paper 10 - fold ($K = 10$) cross-validation has been used [18]. The results of the cross-validation are presented as the average of K scores, as well as the minimum score and standard deviation. This process was performed for every combination of the hyperparameters and was further repeated 10 times for each hyperparameter combination.

2.2.1 Support Vector Machines

SVMs are based on the creation of support vectors, which are used in establishing a hyperplane used to separate instances of different classes in the parameter hyperspace [19]. The benefit of this lays in its use of only those dataset instances closest to the hyperplane, which makes it suitable for application in datasets with a low number of instances - which is why it was selected for this research. In the SVM method we can utilize the vectors written as:

$$\left(\vec{x_i}, y_i\right) \tag{2}$$

where the vector $\vec{x_i}$ is a p dimensional vector, with p being the number of features for the instance i, which must be equal amongst all instances; and y_i represents the class of point $\vec{x_i}$, defined as either 1 or −1 [19]. The hyperplane will be constructed in such a way that it satisfies the condition:

$$\vec{w}\cdot\vec{x_i} - b = 0 \tag{3}$$

with $\vec{w}$ representing the normal vector to the hyperplane and $b/\left\|\vec{w}\right\|$ representing the offset of the hyperplane from the origin in the direction of normal vector $\vec{w}$ [15]. Following that margins are defined. Margins are two supporting planes that separate the two classes (1 and −1), and are defined as $\vec{w}\cdot\vec{x_i} - b = 1$ for class 1 and $\vec{w}\cdot\vec{x_i} - b = -1$, for the class−1. We can define an additional constraint to prevent data points from falling inside the margin. This can be achieved by defining:

$$y_i\left(\vec{w}\cdot\vec{x_i} - b\right) \geq 1, \forall 1 \leq i \leq n \tag{4}$$

where n is the number of data points inside the data set [19]. With these defined, we can define the optimization problem, solving of which for values $\vec{w}$ and b determines the classifier defined as:

$$x \mapsto sgn\left(\vec{w}\cdot\vec{x_i} - b\right) \tag{5}$$

Table 1 Possible hyperparameter values for the SVM algorithm with the total number of tested combinations.

Hyperparameters	Value	Count
C	1.0, 0.95, 0.9, 0.85, 0.8, 0.75, 0.7, 0.65, 0.6, 0.5	10
Kernel	rbf, linear, poly, sigmoid	4
Degree	1, 2, 3, 4, 5	5
Gamma	auto, scale	2
Total hyperparameter variations		400

A hyperplane is defined by those instances of $\vec{x_i}$ nearest to it, which are referred to as "support vectors" [15].

As SVM can only perform binary (two-class) classification the multiclass classification is performed using the "one versus all" principle. Using this principle, multiple classification machines are created, each of which classifies one class as a positive and the instances of all the other classes as the negative class. Then, multiple classifiers are applied to the instance in an attempt to determine which class it belongs to [20].

The hyperparameters used for SVM, in both binary and multi-class classification, are given in Tab. 1.The brief descriptions of the hyperparameters adjusted follow [21]:

- C - regularization parameter, where the strength of regularization is inversely proportional to C,
- kernel - a mathematical function which is used for defining the hyperplane shape,
- degree - degree of the polynomial used for the polynomial kernel (ignored by other kernel types),
- gamma - the coefficient for RBF, polynomial, and sigmoid kernels which adjusts the influence of a single training sample.

2.2.2 Multilayer Perceptron

MLP is a type of artificial neural network (ANN). While commonly needing large amounts of data, MLP has, in some cases, shown good performance when applied to small datasets. MLP consists of an input layer, an output layer, and one or more hidden layers [16]. These layers consist of neurons, connected to the subsequent layer through weighted connections [22, 23].

Table 2 List of possible hyperparameters, along with a total number of possible architectures (hyperparameter combinations)

Hyperparameter	Value	Count
Hidden Layer Sizes	(84, 84, 84, 84), (84, 84, 84), (84, 84), (84), (42, 42, 42, 42), (42, 42, 42), (42, 42) ,(42), (21, 21, 21, 21), (21, 21, 21), (21, 21), (21), (84, 42, 42, 21), (42, 21, 21), (84, 42, 21), (42, 21)	16
Activation Function	'relu', 'identity', 'logistic', 'tanh'	4
Solver	'adam', 'lbfgs'	2
Learning Rate Type	'constant', 'adaptive', 'invscaling'	3
Initial Learning Rate	0.1, 0.01, 0.5, 0.00001	4
L2 Regularization	0.01,0.1,0.001, 0.0001	4
Total Number of hyperparameters		6144

In the input layer, the values of neurons are defined as

the values of inputs from the dataset the vector $\vec{x_i}$ mentioned in the previous section. The values of neurons in the following layers (hidden and output layers) are defined as the activated weighted sum of previous layers outputs.

The weights are initially randomized but are then set through the training process consisting of forward and backward propagation [15]. Forward propagation takes a data point and calculates the output by placing it at the input of the neural network, obtaining the predicted output value $\bar{y}$. This value is then compared to the real output value contained in the dataset y.

In the case of the multiclass classification, the MLP functions by allowing multiple outputs by using multiple neurons in the final layer and classifying the input into the class of the neuron with the highest output value [22]. The number of hyperparameters that can be adjusted in MLP is significantly higher than in SVM, due to which there are a larger number of hyperparameters, and their possible values, to be included in grid search [21]. These values are given in Tab. 2.

As can be seen from Tab. 2 the following hyperparameters were adjusted during the grid search algorithm [21]:
• hidden Layer sizes: A tuple which describes the number of neurons in each layer of the MLP,
• activation function - a function used on the output of each neuron in the MLP,
• learning rate type - a parameter which defines the type of adjustment made to the learning speed of the MLP during the backpropagation process,
• initial learning rate - the initial value of learning rate,
• $L2$ - the regularization parameter which adjusts the influence of the more influential parameters,
• solver - the algorithm used for calculating the new weight values during the backpropagation process.

2.2.3 Convolutional Neural Network

CNNs are neural networks that are based on the process of convolution defined for input x and the filter h. Feature extraction is performed using convolution layers. The values of filter kernels used to play a similar role to the connection weights of the MLP, being adjusted during the training process to minimize the error. In the presented research the input matrix is the one defined by [6]:

$$\begin{aligned} y(m,n) &= x(m,n)\cdot h(m,n) = \\ &= \sum_{i=0}^{15}\sum_{j=0}^{6} x(i,j)\cdot h(m-i,n-j) \end{aligned} \tag{6}$$

In addition to the convolution layers pooling layers are also used. These layers follow the application of a convolution layer and serve as an additional feature extractor, and they are used to lower the amount of data utilized inside the convolution layers [24].

From the above, it can be seen that the architecture of the CNN is defined through the stacking of convolution and pooling layers, along with the activation layers. The internal size of the data is transformed by the application of each layer. The new size, in both horizontal and vertical directions, can be calculated with:

$$W_{\text{new}} = \left[\frac{W-K-2P}{S}\right]+1 \tag{7}$$

where W_{new} is the size after convolution, W is the input size before the convolution, K is the filter kernel size, P is the padding size, and $S = 1$ is the stride.

After the application of convolution, pooling, and activation layers the resulting tensor is flattened into a 1-dimensional vector, which is then used as an input in a two-layer artificial neural network, with a single output neuron and the number of inputs equal to the size of the aforementioned vector. This allows the summation of the CNN outputs and the obtainment of the output value [25]. Due to this shape of the final layer, the multiclass classification can be performed in the same manner as in the case of MLP.

The architecture of the CNN is going to be defined due to the relatively small size of the initial input matrix taken from the dataset, given in Eq. (1). Taking into consideration Eq. (7), and the fact that the output of its per size needs to be larger than 0, we are limited in the number of convolution and pooling layers that can be applied due to the input matrix only having 6 horizontal elements. Because of this, in this research, the architecture of the CNN is predetermined and it consists of a 2-dimensional convolution layer, with 32 filters with a kernel size of (2, 2), followed by an activation layer, and max pooling with the kernel size of (2, 2). This is then followed by another identical convolution layer and activation. Pooling is not used due to the number of horizontal elements being reduced to 1. After flattening, the resulting vector, according to Eq. (7) consists of 64 elements, and uses sigmoidal activation in the one or 13 output neurons; where a single neuron is used for binary classification and 13 neurons are used for multiclass classification.

Due to the architecture being limited by the input data, the modifications within the grid search are only made to the activation function used in layers and the solver used to train the CNN. These values are given in Tab. 3.

Table 3 Varied activation functions

Hyperparameter	Values	Count
First Activation	ReLU, SeLU, Tanh, Identity, ELU, Sigmoid	6
Second Activation	ReLU, SeLU, Tanh, Identity, ELU, Sigmoid	6
Total Hyperparameter Values		36

2.2.4 Siamese Neural Network

SNN is a novel neural network architecture primarily utilized for classification. The differentiating factor in comparison to more commonly used neural networks is that classification is performed by comparing an instance of data of an unknown class, with the instance of data for which the class is known [26]. The value that the network returns is the similarity between the two instances of data, with higher similarity meaning that the two data points belong to the same class.

The first part of training an SNN is adjusting the dataset. This requires the dataset to be transformed in such a way that each data point in the new data set consists of

data pairs [27].

If it can be assumed that a previously used dataset is defined as:

$$D = \left[(M_1, y_1),\ (M_2, y_2),\ldots,\ (M_n, y_n) \right] \tag{8}$$

where $M_k, k \in (0, n)$ represents the measurement matrix defined in Eq. (1), and $y_k, k \in (0, n)$ represents the class the matrix belongs to. The total number of instances in the dataset is n. This dataset is then reformatted into the shape of:

$$D' = \left[\left(M_1^1, M_1^2, y_{M_1^1}, y'_1 \right) \ldots \left(M_n^1, M_n^2, y_{M_n^1}, y'_n \right) \right] \tag{9}$$

As it can be seen, each data point contains two matrices, with M_k^1 being of a known class, defined with the $y_{M_k^1}$, and M_k^2 being the data we want to be classified. The class y'_k marks whether the matrices M_k^1 and M_k^2 belong to the same class, with "1" signifying the equal class, and 0 different classes. This is done by taking all possible combinations of datapoints $d_k = (M_k, y_k)$ in dataset D and defining whether they fall into the same class or not.

To achieve the described process SNN consists of two neural networks. These neural networks are identical and their outputs are connected using a differentiator. The output of this differentiator, $L(o_1, o_2)$, is then used as a loss function during the training of the SNN [28].

In this research, both component neural networks of SNN use the same architecture as CNN described in the previous section. The loss is calculated as the absolute difference between the two outputs, with a lower difference meaning a higher likelihood of the unknown data instance belonging to the same class.

2.3 F_1 score

F_1 score is used as a metric for comparison of the performance of various classifiers. This metric is selected due to its easy definition for multiclass problems, as well as the fact that it provides information on both recall and precision, both of which are important to a task observed in this research.

F_1 score is defined as a harmonic average of recall (R) and Precision (P) [29] with TP representing true positive, FP false positive, and FN false-negative rate.

$$F_1 = \frac{2}{R^{-1} + P^{-1}} = 2 \cdot \frac{R \cdot P}{R + P} = 2 \frac{\frac{TP}{TP + FN} \cdot \frac{TP}{TP + FP}}{\frac{TP}{TP + FN} + \frac{TP}{TP + FP}} \tag{10}$$

In a multiclass classification, the F_1 score is calculated by calculating the individual F_1 scores of each label. This is achieved by transforming a multiclass classification to a binary classification using a so-called "one versus all" system, in which the class for which the F_1 score is calculated is used on its own, while all other classes are treated as a single class. These values are then averaged and the overall F_1 score is calculated as [30]:

$$\overline{F_1} = \frac{1}{13} \sum_{i=1}^{13} F_1^i \tag{11}$$

3 RESULTS

The obtained results are shown in the following tables, with the results for binary classification being given in Tab. 4, and the results for the multiclass classification being given in Tab. 5 for all four algorithms. In the respective tables, the achieved average F_1 score of 10 runs is given, along with the standard deviation. Each score also includes the hyperparameters of the model that achieved those scores.

From the data, it can be seen how high scores can be achieved for binary classification using any of the four methods, with all the algorithms achieving good scores. The SNN achieves the highest scores, also achieving the perfect F_1 score for each of the hyperparameter combinations. Standard deviations show that the SVM provides the least stable best solution, with the highest variation amongst the 10 runs of the K-fold algorithm, with a standard deviation of 0.11. MLP shows a relatively low standard deviation, with CNN achieving an even more stable solution at 0.00027, compared to MLPs standard deviation of 0.01. It should be noted that SNN provides the most stable solution with all the scores achieving the same, perfect, F_1 score.

Table 4 Best results in the case of binary classification

Algorithm	$\overline{F_1}$	σ	Hyperparameters
SVM	0.97273	0.11195	C: 1.0 Degree: 1 Gamma: auto Kernel: rbf
MLP	0.99692	0.01231	HLS: (21,21) AF: ReLU L2: 0.0001 LR: constant LR init.: 0.01 Solver: adam
CNN	0.99859	0.00027	AF1: ReLU AF2: ELU Solver: adam
SNN	1.0	0.0	AF1: ReLU AF2: Tanh Solver: adam

While all four algorithms provide a satisfactory F_1 score, due to the instability of the score within the validation process SVM would not be recommended. With SVM eliminated, the remaining three algorithms (MLP, CNN, and SNN) could be used for classification. SNN positions itself as the natural selection for the algorithm of choice in solving the binary classification of the presented robot fault detection problem, due to achieving perfect scores.

The best results for the binary classification are obtained using SNN, which provides a perfect F_1 score of 1.0, with the best results for multiclass classification being achieved using CNN with the average F_1 score of 0.99.

Table 5 Best results in the case of multiclass classification

Algorithm	$\overline{F_1}$	σ	Hyperparameters
SVM	0.18132	0.27074	C: 1.0 Degree: 1 Gamma: auto Kernel: rbf
MLP	0.46887	0.35713	HLS: (21,21) AF: ReLU L2: 0.0001 LR: constant LR init.: 0.01 Solver: adam
CNN	0.99109	0.01067	AF1: ReLU AF2: ELU Solver: adam
SNN	0.66775	0.03767	AF1: ReLU AF2: Tanh Solver: adam

The scores show a significant drop for multiclass classification, with the only algorithm achieving a satisfactory score being CNN, with the score average F_1 score of 0.99109. Other algorithms achieve very low results, with the best average F_1 scores being 0.19 for SVM, 0.47 for MLP, and 0.67 SNN. It can also be noticed that there are high variations in the results with the same architecture for MLP and SVM, considering the high standard deviation values shown in Tab. 5.

With CNN being the only algorithm that achieves an F_1 score above 0.99, it should be selected as the algorithm for solving the multiclass portion of the presented robot fault classification problem. The inability to achieve a higher score can be explained due to the unbalanced dataset used for fault classification, which can be seen in Fig. 1.

The solution to the composite problem would be a hybrid solution that uses an SNN to detect the problem according to force and torque measurements. SNN, due to the extremely high score, presents itself as a good solution for this part, due to the high importance of detecting that fault has happened within the robotic manipulator operation, so the operation itself can be stopped immediately. If false positives existed they could lower the operational hours of the robotic manipulator, while the false negatives (failing to detect the fault) could cause costly damages to the robotic manipulator and other parts of the manufacturing chain which surround it.

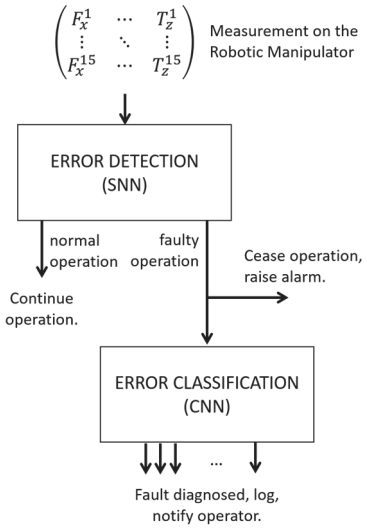

Figure 2 The proposed hybrid model.

If force and torque measurements are classified as faulty operations, they are passed on to the trained CNN algorithm which detects the type of the problem.

While not having as high of the score, CNN still achieves good enough scores for the type of fault being detected. As this information is something that could be provided to the operators, classifying the faults correctly is not as crucial as detecting their presence. This hybrid system is shown in Fig. 2.

4 CONCLUSION

The authors present the issue of detecting and diagnosing the faults of a robotic manipulator. Using a public dataset and multiple AI classification algorithms, the results show the possibility of using such algorithms to provide the system needed to combat the presented issue. It can be seen that those algorithms achieve good results, even with raw, unprocessed data, eliminating the need for potentially costly (in financial and time sense) industrial equipment and algorithms needed to extract information from the data. Due to different algorithms providing the best results, the authors propose the use of a hybrid system that uses an SNN algorithm for the detection of the fault and a CNN algorithm for its classification. The SNN algorithm achieves the goal stated by the authors, of having the perfect F_1 score which is necessary for the implementation of such a solution; with the CNN algorithm achieving not perfect, but satisfactorily high scores, for fault classification.

Future work will concentrate on feature extraction, as presented in the original paper relating to the dataset, to raise the classification quality [6], especially in the multiclass fault detection. Testing of other novel machine learning algorithms may also be a part of future work. Most importantly, building a larger and better-balanced dataset, based on measurements from not just a single, but multiple industrial robotic manipulators, may be crucial for achieving even better, and more robust, results that can be applied generally to robotic manipulators.

Acknowledgments

This research has been (partly) supported by the CEEPUS network CIII-HR-0108, European Regional Development Fund under the grant K.01.1.1.01.0009 (DATACROSS), project CEKOM under the grant KK.01.2.2.03.0004, CEI project "COVIDAi" (305.6019-20), project Metalska jezgra Čakovec (KK.01.1.1.02.0023) and University of Rijeka scientific grant uniri-tehnic-18-275-1447

5 REFERENCES

[1] Xiao, B., Cao, L., Xu, S., & Liu, L. (2020). Robust Tracking Control of Robot Manipulators with Actuator Faults and Joint Velocity Measurement Uncertainty. *IEEE/ASME Transactions on Mechatronics*. https://doi.org/10.1109/tmech.2020.2975117

[2] Baressi Šegota, S., Anđelić, N., Lorencin, I., Saga, M., & Car, Z. (2020). Path planning optimization of six-degree-of-freedom robotic manipulators using evolutionary algorithms. *International Journal of Advanced Robotic Systems*, *17*(2).

https://doi.org/10.1177/1729881420908076
[3] Murphy, R. R. (2019). *Introduction to AI robotics*. MIT press. https://doi.org/10.1108/ir.2001.28.3.266.1
[4] Wei, Y., Hiraga, M., Ohkura, K., & Car, Z. (2019). Autonomous task allocation by artificial evolution for robotic swarms in complex tasks. *Artificial Life and Robotics*, *24*(1), 127-134. https://doi.org/10.1007/s10015-018-0466-6
[5] Wei, Y., Nie, X., Hiraga, M., Ohkura, K., & Car, Z. (2019). Developing end-to-end control policies for robotic swarms using deep Q-learning. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, *23*(5), 920-927. https://doi.org/10.20965/jaciii.2019.p0920
[6] Lopes, L. S. & Camarinha-Matos, L. M. (1998). Feature transformation strategies for a robot learning problem. *Feature Extraction, Construction and Selection*, 375-391. Springer, Boston, MA. https://doi.org/10.1007/978-1-4615-5725-8_23
[7] Costa, M. A., Wullt, B., Norrlöf, M., & Gunnarsson, S. (2019). Failure detection in robotic arms using statistical modeling, machine learning and hybrid gradient boosting. *Measurement*, *146*, 425-436. https://doi.org/10.1016/j.measurement.2019.06.039
[8] Pinto, R. & Cerquitelli, T. (2019). Robot fault detection and remaining life estimation for predictive maintenance. *Procedia Computer Science*, *151*, 709-716. https://doi.org/10.1016/j.procs.2019.04.094
[9] Cheng, F., Raghavan, A., Jung, D., Sasaki, Y., & Tajika, Y. (2019, June). High-accuracy unsupervised fault detection of industrial robots using current signal analysis. *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*, 1-8. https://doi.org/10.1109/icphm.2019.8819374
[10] Mitrevski, A. & Plöger, P. G. (2019). Data-Driven Robot Fault Detection and Diagnosis Using Generative Models: A Modified SFDD Algorithm. https://doi.org/10.1016/b978-0-12-819164-4.00003-0
[11] Piltan, F., Prosvirin, A. E., Sohaib, M., Saldivar, B., & Kim, J. M. (2020). An SVM-Based Neural Adaptive Variable Structure Observer for Fault Diagnosis and Fault-Tolerant Control of a Robot Manipulator. *Applied Sciences*, *10*(4), 1344. https://doi.org/10.3390/app10041344
[12] Yu, C., Cai, Z., Pham, H., & Pham, Q. C. (2019). Siamese Convolutional Neural Network for Sub-millimeter-accurate Camera Pose Estimation and Vision Servoing for Fine Robotic Assembly. https://doi.org/10.1109/iros40897.2019.8967925
[13] Chang, S., Zhang, F., Huang, S., Yao, Y., Zhao, X., & Feng, Z. (2019). Siamese Feature Pyramid Network for Visual Tracking. *2019 IEEE/CIC International Conference on Communications Workshops in China (ICCC Workshops)*, 164-168. https://doi.org/10.1109/iccchinaw.2019.8849954
[14] Oğul, B. B., Gilgien, M. F., & Şahin, P. D. (2019, November). Ranking Robot-Assisted Surgery Skills Using Kinematic Sensors. *European Conference on Ambient Intelligence*, 330-336. Springer, Cham. https://doi.org/10.1007/978-3-030-34255-5_24
[15] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media. https://doi.org/10.1007/bf02985802
[16] Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning*, *1*(2). Cambridge: MIT press. https://doi.org/10.1007/s10710-017-9314-z
[17] Car, Z., Baressi Šegota, S., Anđelić, N., Lorencin, I., & Mrzljak, V. (2020). Modeling the Spread of COVID-19 Infection Using a Multilayer Perceptron. *Computational and Mathematical Methods in Medicine*, *2020*. https://doi.org/10.1155/2020/5714714
[18] Lorencin, I., Anđelić, N., Mrzljak, V., & Car, Z. (2019). Genetic algorithm approach to design of multi-layer perceptron for combined cycle power plant electrical power output estimation. *Energies*, *12*(22), 4352. https://doi.org/10.3390/en12224352
[19] Wang, M. & Chen, H. (2020). Chaotic multi-swarm whale optimizer boosted support vector machine for medical diagnosis. *Applied Soft Computing*, *88*, 105946. https://doi.org/10.1016/j.asoc.2019.105946
[20] Cheng, Y., Zhu, H., Hu, K., Wu, J., Shao, X., & Wang, Y. (2019). Multisensory data-driven health degradation monitoring of machining tools by generalized multiclass support vector machine. *IEEE Access*, *7*, 47102-47113. https://doi.org/10.1109/access.2019.2908852
[21] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *Journal of machine Learning research*, *12*, 2825-2830.
[22] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer. https://doi.org/10.1016/c2009-0-22409-3
[23] Baressi Šegota, S., Lorencin, I., Anđelić, N., Mrzljak, V., & Car, Z. (2020). Improvement of Marine Steam Turbine Conventional Exergy Analysis by Neural Network Application. *Journal of Marine Science and Engineering*, *8*(11), 884. https://doi.org/10.3390/jmse8110884
[24] Lorencin, I., Baressi Šegota, S., Anđelić, N., Blagojević, A., Šušteršić, T., Protić, A., & Car, Z. (2021). Automatic Evaluation of the Lung Condition of COVID-19 Patients Using X-ray Images and Convolutional Neural Networks. *Journal of Personalized Medicine*, *11*(1), 28. https://doi.org/10.3390/jpm11010028
[25] Lorencin, I., Anđelić, N., Mrzljak, V., & Car, Z. (2019). Marine objects recognition using convolutional neural networks. *Nase more, 66*(3), 112-119. https://doi.org/10.5194/egusphere-egu2020-17431
[26] Chicco, D. (2020). Siamese neural networks: An overview. *Artificial Neural Networks*, 73-94. https://doi.org/10.1007/978-1-0716-0826-5_3
[27] Vargas, C., Zhang, Q., & Izquierdo, E. (2020). One Shot Logo Recognition Based on Siamese Neural Networks. *Proceedings of the 2020 International Conference on Multimedia Retrieval*, 321-325. https://doi.org/10.1145/3372278.3390734
[28] Hayale, W., Negi, P., & Mahoor, M. (2019). Facial Expression Recognition Using Deep Siamese Neural Networks with a Supervised Loss function. *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, 1-7. IEEE. https://doi.org/10.1109/fg.2019.8756571
[29] Wang, R. & Li, J. (2019). Bayes Test of Precision, Recall, and F1 Measure for Comparison of Two Natural Language Processing Models. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4135-4145. https://doi.org/10.18653/v1/p19-1405
[30] Wang, R., Fan, J., & Li, Y. (2020). Deep Multi-Scale Fusion Neural Network for Multi-Class Arrhythmia Detection. *IEEE journal of biomedical and health informatics*, *24*(9), 2461-2472. https://doi.org/10.1109/jbhi.2020.2981526

Contact Information:

Sandi BARESSI ŠEGOTA, mag. ing. comp.
(Corresponding Author)
Faculty of Engineering, University of Rijeka,
Vukovarska 58, 51000 Rijeka
E-mail: sbaressisegota@riteh.hr

Nikola ANĐELIĆ, mag. ing. mech.
Faculty of Engineering, University of Rijeka
Vukovarska 58, 51000 Rijeka
E-mail: nandelic@riteh.hr

Zlatan CAR, dr. sc. dipl. ing.
Faculty of Engineering, University of Rijeka
Vukovarska 58, 51000 Rijeka
E-mail: car@riteh.hr

Mario Šercer, dipl. ing. dr. sc.
Razvojno edukacijski centar za metalsku industriju Metalska jezgra Čakovec
Bana Josipa Jelačića 22 D, 40 000 Čakovec
E-mail: mario.sercer@mev.hr