Towards a Trade-off Between Accuracy and Computational Cost for Embedded Systems: a Tactile Sensing System for Object Classification

Youssef Amin, Christian Gianoglio, and Maurizio Valle

Department of Electrical, Electronic, Telecommunication Engineering and Naval Architecture - DITEN, University of Genoa, Italy
{youssef.amin, christian.gianoglio}@edu.unige.it, maurizio.valle@unige.it

Abstract. The deployment of the inference phase in self–standing systems, which have resource–constrained embedded units, is faced with many challenges considering computational cost of the elaboration unit. Therefore, we propose using a learning strategy based on a loss function that leads to finding the best configuration of the prediction model balancing the generalization performance and the computational cost of the whole elaboration system. We validate our proposal by integrating a tactile sensing system on a Baxter robot to collect and classify data from five daily–life objects using four different algorithms. Results show that the best performance, when the computational cost is not relevant, is achieved by the fully–connected neural network using 16 features, while, when the computational cost matters, the loss function showed that the kernel SVM with 4 features has the best performance.

Keywords: Tactile sensing systems, object classification, artificial intelligence for embedded devices, Computational load.

1 Introduction

In recent decades, many research works have been focused on deriving object properties from tactile signals – using Machine Learning (ML) – to differentiate between different daily–life objects. Several studies have adopted the human approach for tactile object identification, the 'exploratory procedures' (EPs) [1], where typical movement patterns are carried out to collect information about objects. However, such approach demands a wide range of tactile data processing, thus it increases the work complexity. Furthermore, new technologies have emerged such as tactile sensing systems that consist of tactile sensors and embedded electronics. These embedded units of such systems are required to perform signal processing, feature extraction, and prediction. Usually, ML algorithms are utilized to perform the classification. However, the execution of all these tasks can be computationally very expensive for resource − constrained devices. Knowing that each of these tasks affects the predictor performance and the computational cost, it is important to find the best configuration (i.e. signal processing techniques, extracted features, model architecture, etc.,) for the elaboration unit, that trades–off between the computational cost and the predictor performance, before deploying the elaboration unit into a prior designed embedded system.

In this paper we present the implementation of a learning strategy based on the evaluation of a loss function [2], in order to find the best configurations of the elaboration unit stages, addressing the classification of 5 objects based on tactile data. The learning strategy provides a balance between the computational cost of the whole elaboration unit (i.e. in terms of FLOPs) and the generalization performance of the predictors. In this regard, we apply an automated technique to remove unneeded information from a single unplanned grasping action (inspired from the concept of haptic glance found in humans [3]) and extract features, to reduce the computational cost on the elaboration unit. Moreover, for prediction, we implement ML algorithms that have proven to be suitable for resources–constrained devices [4, 5].

The remainder of the paper is: Section 2 presents related works, Section 3 describes our proposal, Section 4 explains the experimental setup and the workflow, Section 5 presents the results, eventually Section 6 concludes the paper.

2 Related Works

Before a few decades, The awareness of robotic systems to their surrounding environment was confined by visual exploration, which helps to understand the workplace for better manipulation. However, to acquire certain object properties (i.e. hardness, texture, weight, etc.,) physical interaction with objects is required, and in some cases manipulation is needed. Furthermore, ML and deep learning algorithms have been utilized to extract high–level information from tactile data. Some of these studies have aimed to classify texture [6], stiffness [7, 8], or objects recognition [9–11]. The authors in [12, 7] integrated a tactile sensing system into a Baxter robot to collect data from three objects of different hardness. They used for the classification a single layer feed-forward neural networks (SLFNNs) that proved to be effective in terms of generalization performance and the computational cost of the predictor. Similarly in [13], the authors proposed a methodology to employ and then optimize classifiers of varying complexity using a principle optimization algorithm. The later algorithm, allowed to automatically configure the coarse–to fine–networks, as it achieves a trade–off between energy consumption and accuracy for the inference. As a result for that, the optimized corase–to–train network reduced the overall energy for image classification problem without any loss in accuracy. The authors in [9] utilized a four–fingered hand that compresses 241 tactile skin sensors distributed over the hand, equipped on TWENDY–ONE robot. The data was collected by grasping 20 different daily–life objects, of different shapes and hardness. For object recognition, a deep learning neural network approach was employed along with a denoising auto–encoder (DAE) and a dropout. In addition, two feature learning techniques: the self–organizing maps and the principal component analysis were applied. For the comparison, a single layer neural network was implemented and the recognition rates for different sensor modalities were tested. whereas in [10], the authors evaluated the performance of different neural network–based models, namely Convolutional Neural Networks (CNN) and Long–Short Term Memory (LSTM), on an object recognition task for 9–classes and compare the results on tactile grasping data collected using two different fingertip tactile robot sensors. Experimental results show that the LSTM–based model outperforms the CNN models. They also present a method to extend the available data for training, leading to performance improvement. Authors in [11] proposed a hybrid methodology for performing tactile classification and feature extraction, having a row of TakkTile tactile pressure sensors, integrated into an under–actuated robotic hand. The setup was used to collect data while performing a single unplanned grasp. However, exploratory motions were not applied. Features were extracted from the sensors readings and the actuator position, at three instances of the grasping process, and employed for object classification using the random forest method. This approach provided a system with low computation and complexity overheads for haptic sensing applications. On the other hand, the authors proposed in [8] a semi–supervised generative adversarial network (GAN) for hardness detection to mitigate the expensive process of data labeling. Unsupervised training of GAN with a large number of unlabeled samples provides the architecture and initial parameter values for the hardness recognition network (HRN), which is trained with manually labeled samples corresponding to each hardness level. The hardness detection result can be determined online by importing the tactile data collected by the robotic forearm into the trained HRN.

In these works, the authors did not evaluate the computational cost of their models nether their elaboration unit, except in [7, 13]. However, it is crucial to keep track of the computational cost along with the generalization accuracy knowing that high model accuracy can be obtained at the expense of higher computations. In this context, we compute the FLOPs for all the operations starting from signal processing (i.e. filtering of the tactile signal) until the prediction of object. We also employ a loss function in the learning strategy to obtain the best model configuration balancing between the accuracy and the computational cost for the whole elaboration unit.

3 Proposal

A tactile sensing system is made of a sensing patch and an elaboration unit. The elaboration unit includes different stages: signal processing, feature extraction, and prediction. All these stages require a certain amount of power consumption which may form a problem in resource–constrained devices. On the other hand, having a good model with a high generalization performance sometimes come with many expenses on the computational cost. Therefore, balancing the trade–off between generalization performance and computational cost becomes crucial. We propose using a learning strategy based on a loss function that balances the generalization performance and the computational cost, keeping into consideration all the stages of the elaboration unit.

3.1 Loss Function

In [2], the authors proposed a loss function that during the training phase is capable of selecting the best model in terms of the trade–off between the predictor accuracy and its computational cost. Procedure 1 in [2] depicts the learning strategy that we adopt in this paper as well. It consists of a data–driven approach based on the out–of–sample technique [14] where the original dataset is split into training and validation sets. The training set is used to train the models on all the possible configurations of the hyper–parameters, solving (2). The validation set is then employed to find the best model minimizing the loss function, solving (1). The loss function consists of two terms:

$$i^* = \operatorname*{argmin}_i \hat{L}_m(f^*_{n,i}) + \theta R_H(f^*_{n,i}) \tag{1}$$

where

$$f^*_{n,i} = \operatorname*{argmin}_{f \in \mathcal{F}_i} \hat{L}_n(f) + \lambda R(f). \tag{2}$$

In (2), $\mathcal{F}_i$ represents one of the models indexed by the hyper–parameters, $\hat{L}_n(f)$ is the empirical risk on the training set, and $\lambda R(f)$ describes the regularizer with its weight. In (1), $\hat{L}_m$ denotes the empirical risk on the validation set, while the second term θR_H is the hardware constraint weighted by a parameter that balances the trade–off between accuracy and computational. Adopting a value of $\theta = 1$ means giving the same importance to the hardware constraint and the accuracy, while with a value greater than 1 the loss function promotes lighter model (e.g. model with lower size, lower number of parameters, lower FLOPs, etc.) with respect of achieving the best accuracy. Therefore, Procedure 1 is employed taking into account different combinations of the elaboration unit stages. In the following, we describe in detail the stages of the elaboration unit and their computational cost in terms of FLOPs.

3.2 Signal conditioning

The tactile sensing system employed for this study is the same used in [7], which consists of a low–cost piezoelectric sensing array and a low–power interface electronics (IE) that has a low–current input analog–to–digital converters and is able to acquire data from multiple input channels (i.e. 32). We employ a method to reduce the number of samples acquired from each channel, by introducing a threshold that triggers a real–time extraction of Ns number samples during a grasp. Ns represents the length of the generated signal. Finally, we filter the generated signal $\tilde{\boldsymbol{x}}$ using a two–samples Moving Average filter (MA): $y[i] = \alpha\tilde{x}[i] + (1-\alpha)y[i-1]$; both α and Ns are user defined parameters. Based on [15], the computational cost of filtering with MA is $(3Ns + 5)$ FLOPs.

3.3 Feature Extraction

Similarly to [7], we extract as features the statistical moments from the filtered signals $\boldsymbol{y}$. They proved to be efficient from the computational cost side and led to a high accuracy during the prediction stage. Table 1 reports the number of FLOPs for each feature. We considered that is possible to employ partial results of a feature to compute the others. As an example, the mean and standard deviation of a signal can be used in the computation of the skewness, thus we save FLOPs.

Table 1. Extracted Features and FLOPs Count

Extracted Features	**FLOPs Count**
Mean	$Ns + 1$
STD	$3Ns + 5$
Skewness	$2Ns + 7$
Kurtosis	$2Ns + 5$

3.4 Predictors

We present four different ML algorithms to classify the five daily objects. One of the algorithms is based on the support vector machine (i.e. SVM), which aims to classify different classes by finding the hyper–planes that maximize the margin between two classes. In this work, a Gaussian kernel (i.e., the radial basis function RBF) is used with the SVM (K–SVM). The hyper–planes of SVM classifier are represented by the following function for a two–class problem: $f(\boldsymbol{z}) = \sum_{i=1}^{nSV} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{z}) + b$; where $\boldsymbol{z}$, nSV, α_i, y_i, $\boldsymbol{x}_i$, K, and b denote the tested datum, the number of support vectors, the coefficient of the support vector, the corresponding label, the training datum that lies on the support vector, the kernel, and the bias, respectively. More Information about SVM are in [16].

The other three algorithms are neural networks made of one hidden layer, thus recognized as single–layer feed–forward neural networks (SLFNNs). The SLFNNs are known to be able to provide high generalization performance in classification while maintaining the computational cost low. The general prediction function of a SLFNN for a two–class problem is: $f(\boldsymbol{z}) = \sum_{i=1}^{N} \beta_i \phi(\boldsymbol{z} \cdot \boldsymbol{\omega}_i + b_i)$; where $\boldsymbol{z}$ the tested datum, N is the number of hidden neurons, β_i the weights between the hidden and output layers, ϕ the activation function, $\boldsymbol{\omega}$ the weights between the input and hidden layers, and b_i the neuron bias. The first SLFNN is the fully–connected neural network that employs ReLU activation functions (R–FC), and is trained by the backpropagation technique. In contrast, the second and third SLFNNs are based on the Extreme Learning Machine (ELM) paradigm [17]. In ELM, parameters $\boldsymbol{w}, b$ are set randomly, and training demands solving a Regularized Least Square (RLS) problem to learn the beta connections. For the first ELM network, we assigned a ReLU activation function (R–ELM). While for the second network, we adopted a hard limit activation function (H–ELM) and set the $\boldsymbol{w}$ weights as a power of two due to its effectiveness on low power and low cost devices [18], [4].

Table 2 shows the FLOPs count of each of the four algorithms, where nf, nc, N, and nSV correspond to the number of features, number of classes, number of neurons, and number of support vectors. In case of a two–class classification problem $nc = 1$.

Table 2. SLFNNs and FLOPs Count

SLFNNs	**FLOPs Count**
H–ELM	$N(2nf + 1) + nc * N$
R–ELM	$N(2nf + 2) + 2nc * N$
R–FC	$N(2nf + 2) + 2nc * N$
K–SVM	$\sum_{i=1}^{nc} nSV_i * (3nf + 10) + 1$

4 Experimental setup

To evaluate the performance of different model architectures to classify grasped objects –by evaluating the loss function (1)– we integrated our tactile sensing system on the gripper of Baxter arm. The system comprises a P(VDF–TrFE) piezoelectric sensing patch and IE, installed on one side of the gripper, to acquire tactile data while running grasping experiments on five daily–life objects.

4.1 System setup

Shown in Figure 1 the system workflow. Similar to the previous work in [7], we choose the same tactile sensing patch, which has high sensitivity over a wide frequency bandwidth (0.5Hz–1kHz). This patch contains a matrix of eight piezoelectric sensors (4 x 2 = 8 sensors) uniformly distributed, forming a rectangular sensing area of $2.1 \times 1.1cm^2$. As well, it has a spatial resolution of $1cm$ center–to–center pitch and a sensor diameter of $2mm$. Before doing any experiment, the patch is shielded –using a special conductive tape– to carry noises (i.e. external charges) to the ground. In addition, a substrate and a thin protective layer is added to the bottom and top side of the patch, respectively. we use the IE to acquire tactile data from sensors. The IE contains an ARM–cortex M0 micro–controller that leads to sample the signals on 32 channels with a frequency of 2KSamples/sec. To integrate our sensing patch into the Baxter robot, we designed a 3D printed gripper fingers that perfectly fit the sensing patch (see Figure 2a). On the other hand, we fasten the low–power IE to the end of the Baxter arm using an elastic fabric strap. The IE is set to transmit continuously filtered tactile data to the host PC through a USB connection. The data collection process is controlled using a LabVIEW GUI ,which we designed for reading, visualizing, and saving the tactile data from eight sensors. Eventually, automatic extraction of grasp peak from each tactile signal, followed by features extraction, then the training of models is done offline using the PC (Figure 1).

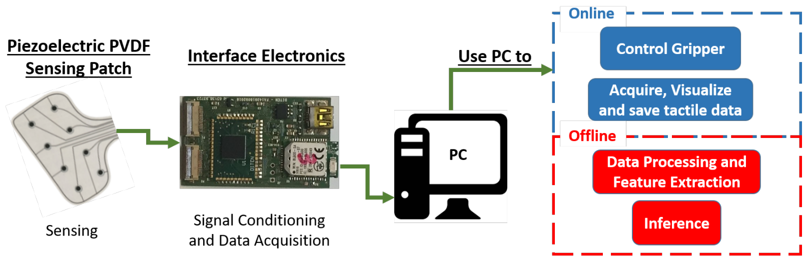
Piezoelectric PVDF Sensing Patch
Interface Electronics
Use PC to
Online
Control Gripper
Acquire, Visualize and save tactile data
Offline
Data Processing and Feature Extraction
Inference
PC
Sensing
Signal Conditioning and Data Acquisition

Fig. 1. System Workflow.

4.2 Objects

For the experiments, we select a set of five different 'daily life' objects shown in Figure 2b. These objects have three types of surfaces: flat (e.g. one side of the eyeglasses case and the cubic box), circular (e.g. the cola can and the tennis ball), and curved (e.g. the shampoo bottle and the other side of the eyeglasses case) surfaces. The objects are different in shape, materials, and size. But, they all have a good impact resistance (i.e. high stiffness) and are big enough to be in contact with many sensors when grasped.

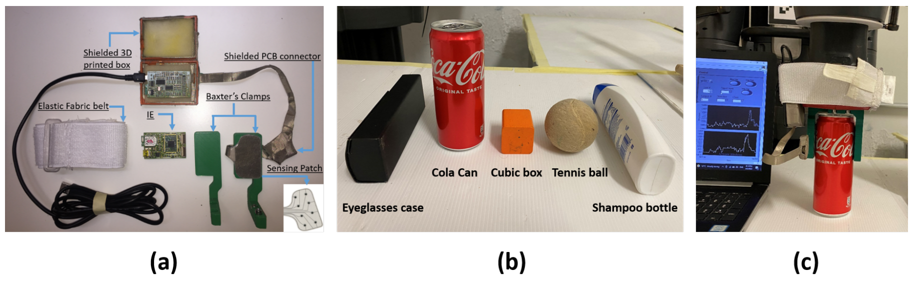

Fig. 2. a) System Setup. b) Set of objects used. c) Example of object grasping.

4.3 Data collection

Data was collected by grasping each of the five objects 400 times, at different positions, using the Baxter gripper as shown in Figure 2c. Our experiments do not involve any post–grasp manipulation; therefore, the arm position is fixed. Before the experiments, we determine the closure position of the gripper for each object, which corresponds to the gripper position when the touch occurs. We set the gripper velocity and force to $v = 5cm/s$ and $F = 0.03 * Fmax$ $(Fmax = 35N)$. The chosen objects are hard enough to avoid any deformation when grasped. During each trial, the gripper closes –at a constant velocity and force– to the predefined position, and remains closed for 1.2 seconds; then, it opens for two seconds. The trials are repeated with no sensors and/or gripper feedback (open–loop control). Grasping control is done by the robotic operating system (ROS). The IE continuously acquires the tactile data from all sensors simultaneously at a high sampling frequency (2KSps) and transmits them to the PC. According to Section 3.2, the signals are pre–processed before extracting the features. Using Matlab, we automatically extract 150 samples from the grasp peak by taking 25 samples before the sample that has the minimum value and 125 samples after. This process is done for all the channels simultaneously, and for all the trials.

Eventually, we build six datasets based on the combination of the number of sensors (i.e., excluding four sensors or using all of them) and the extracted features. Each dataset contains 2000 data, i.e. 400 samples for each class. Table 3 gives the details of the six datasets. Each column corresponds to a dataset, the first row reports the name of the datasets, while the second, third, fourth, and the fifth rows report the number of sensors used to build the datasets, the number of features extracted from each sensor, the type of feature (i.e. Mean, standard deviation=STD, kurtosis=Kur, and skewness=Skew), and the total number of features, respectively. The six datasets correspond to six different feature extraction stages. The loss function will be evaluated to find the best combination of features extraction stage and predictor.

Table 3. Generated Datasets

Datasets Names	4S1F	4S2F	4S4F	8S1F	8S2F	8S4F
Number of Sensors	4	4	4	8	8	8
Number of Features per Sensor	1	2	4	1	2	4
Type of features	Mean	Mean STD	Mean,STD Skew, Kur	Mean	Mean STD	Mean,STD Skew, Kur
Total Num of Features	4	8	16	8	16	32

4.4 Training strategy

Four algorithms are built to classify the five objects, therefore the SLFNNs have five output neurons while the K–SVM presents five linear separators by applying the 'one–vs–one' approach. In the SLFNNs, the Softmax activation function assigns the label to the neuron that has the highest probability. In the K–SVM, the label of the tested datum was assigned by evaluating its position with respect to all the separators. All the algorithms were trained over the six datasets previously described. The SLFNNs hyper–parameters are defined as follows: the hidden neurons $N = \{50, 100, 200, 300\}$, the regularization term L2 $\lambda = \{10^i,\ i = -4, -3, ..., 5\}$, random weights and biases between $[-1, 1]$ (for

H–ELM they were represented as the power of 2 [18, 4]); while for the K–SVM: the RBF kernels parameter $\gamma = \{10^i,\ i = -4, -3, ..., 5\}$, and same regularization term λ as for SLFNNs. For the FC, the number of epochs, batch size, patience on the validation accuracy for the early stop criterion, and the learning rate using the Adam optimizer were fixed to 200, 64, 20, and 0.01, respectively. The loss function (1) is evaluated by setting the parameter θ as $\theta = \{0, 0.5, 1, 2, 5, 10\}$. In this way, $\theta = 0$ means that only the accuracy is relevant during the evaluation, while the greater the θ the higher the relevance of having a light model with a low computational cost. For all classifiers, we apply a stratified K–folds cross–validation with $K = 10$: the datasets are divided into 10 folds which, in turn, 9 contribute to form the training set and 1 the test set. Therefore, each algorithm is trained 10 times on each dataset (i.e., 10 runs). The stratification leads to having the same proportion of data for each class in the folds with respect to the original dataset. In each run, the training data are randomly split in training/validation sets using the 75% of data as training and 25% as validation. The validation set is employed to evaluate the loss function for each θ value. Eventually, the test set is used to assess the generalization performance of the best models in terms of accuracy.

5 Results

Through the evaluation of the loss function, it is possible for each algorithm to find the best set of hyper–parameters and the best features set that leads to the best compromise between generalization accuracy and computational cost measured as the number of FLOPs, for all the values of the θ parameter. The FLOPs are considered along all the elaboration stages and normalized between 0 and 1. The number of FLOPs and generalization accuracies are averaged along the 10 folds, for each pair features set–algorithms and for each value of θ.

Procedure 1 in [**?**] is evaluated on each algorithm to find the best combination of features set and predictor hyper–parameters, for all the values of θ. Then, we compute the loss function (1) for each θ on the test data. Results are reported in Figure 3.In the figure, the loss values are multiplied by 100 for a better description. For each theta, there are four colored bars that correspond to the following algorithms: K–SVM, R–FC, R–ELM, and H–ELM. For $\theta = 0$ (meaning that the computational cost is not relevant), the best loss function value is obtained by R–FC (99.6), followed by R–ELM (98.0), K–SVM (97.6), and finally H–ELM (97.2). In this case, the score corresponds to the test accuracy. However, for $\theta > 0$, the loss function decreases, and different results are achieved as the K–SVM outperforms the others. Between $\theta = 0.5$ and $\theta = 10$ the loss function decreases slightly in K–SVM (≈5%), and significantly in R–FC (≈20.3%), R–ELM (≈20.5%), and H–ELM (≈24%). As a result, the best loss function is always achieved by the K–SVM. For all $\theta > 0$, the H–ELM has the lowest values; which is normal because it requires much more neurons than R–ELM and R–FC to attain similar generalization performances in terms of accuracies [7].

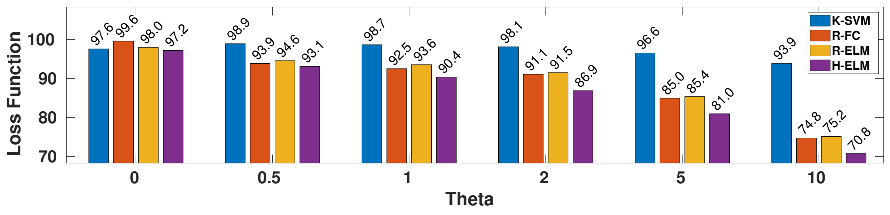

Fig. 3. The Loss function of the four algorithms, at different θ weights.

Table 5 presents the best configurations chosen by the evaluation of the loss function, showing the number of sensors and extracted features, based on each algorithm and θ value. In the table, the columns from the left to the right reports, respectively, the following: algorithms, θ, number of sensors, extracted features, average FLOPs, and the average accuracy on the test set. For $\theta > 0$, we obtained the same dataset (i.e. 4F1S) for all algorithms; this dataset utilizes the minimal number of sensors (i.e. 4) and type of features (i.e. the mean M). Whereas, for $\theta = 0$, the best configuration for R–ELM (i.e. 8 sensors, one type of features) is different from the other algorithms (having 4 sensors and 2 types of features: M and STD). Moreover, for all algorithms, we observe that: the bigger the θ lower the averaged computational cost and accuracy, except in the case of K–SVM; nevertheless, it gives the highest accuracies with the lowest computational costs for $\theta > 0$. And, the R–FC attains the best accuracy for $\theta = 0$. Therefore, by employing the loss function in the learning strategy for each algorithm, not only did we find the configurations which provide the best trade–off between the generalization accuracy and number of FLOPs, but also the best predictor (K–SVM) for object classification when the computational cost is relevant.

6 Conclusion

This work presents a learning strategy based on a loss function to find the best configurations of algorithm, when balancing between the predictor accuracy and the computational cost of the whole elaboration unit (i.e. signal processing, feature extraction, and prediction). A parameter in the loss function is tuned to weight the importance of the computational cost. The loss function is employed for the classification of five objects, measuring the computational cost in terms of FLOPs. Results show that the best performance, when the computational cost is not relevant, is achieved by the fully–connected neural network with the ReLu activation function using 16 features while, when the computational cost matters, the loss function chooses the kernel SVM of 4 features as the best configuration. In the future work the model with the best trade-off can be chosen to be deployed on a resource-constrained devices.

Table 4. Best Configurations showing the number of sensors (NS), extracted features, average FLOPs, and average generalization accuracy.

Algorithms	θ	NS	Features	Av. FLOPs	Av. Accuracy STD
K–SVM	0	8	M&STD	9855	97.60 ±3.38
	0.5	4	M	2818	99.21 ±1.69
	1	4	M	2818	99.21 ±1.69
	2	4	M	2818	99.21 ±1.69
	5	4	M	2818	99.21 ±1.69
	10	4	M	2818	99.21 ±1.69
R–FC	0	8	M&STD	9300	99.60 ±1.27
	0.5	4	M	3296	95.20 ±3.16
	1	4	M	3296	95.20 ±3.16
	2	4	M	3156	95.20 ±3.16
	5	4	M	3156	95.20 ±3.16
	10	4	M	3156	95.20 ±3.16
R–ELM	0	8	M	5770	98.01 ±2.83
	0.5	4	M	3156	95.60 ±3.51
	1	4	M	3156	95.60 ±3.51
	2	4	M	3156	95.60 ±3.51
	5	4	M	3156	95.60 ±3.51
	10	4	M	3156	95.60 ±3.51
H–ELM	0	8	M&STD	9620	97.20 ±2.70
	0.5	4	M	3646	95.20 ±4.14
	1	4	M	3506	94.00 ±5.42
	2	4	M	3226	91.60 ±6.39
	5	4	M	3156	91.21 ±7.01
	10	4	M	3156	91.21 ±7.01

Acknowledgment

The authors acknowledge financial support from TACTIle feedback enriched virtual interaction through virtual realITY and beyond (Tactility) project: EU H2020, Topic ICT-25-2018-2020, RIA, Proposal ID 856718.

References

1. Lederman, S.J., Klatzky, R.L.: Hand movements: A window into haptic object recognition. Cognitive Psychology 19(3), 342–368 (1987)
2. Gianoglio, C., Ragusa, E., Gastaldo, P., Valle, M.: A novel learning strategy for the trade-off between accuracy and computational cost: A touch modalities classification case study. IEEE Sensors Journal 22(1), 659–670 (2021)
3. Klatzky, R.L., Lederman, S.J.: Identifying objects from a haptic glance. Perception & Psychophysics 57(8), 1111–1123 (1995)
4. Ragusa, E., Gianoglio, C., Zunino, R., Gastaldo, P.: A design strategy for the efficient implementation of random basis neural networks on resource-constrained devices. Neural Processing Letters pp. 1–19 (2019)

5. Ragusa, E., Gianoglio, C., Zunino, R., Gastaldo, P.: Random-based networks with dropout for embedded systems. Neural Computing and Applications 33(12), 6511–6526 (2021)
6. Chun, S., Kim, J.S., Yoo, Y., Choi, Y., Jung, S.J., Jang, D., Lee, G., Song, K.I., Nam, K.S., Youn, I., Son, D., Pang, C., Jeong, Y., Jung, H., Kim, Y.J., Choi, B.D., Kim, J., Kim, S.P., Park, W., Park, S.: An artificial neural tactile sensing system. Nature Electronics 4(6), 429–438 (Jun 2021), https://doi.org/10.1038/s41928-021-00585-x
7. Amin, Y., Gianoglio, C., Valle, M.: Computationally light algorithms for tactile sensing signals elaboration and classification. In: 2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS). pp. 1–6 (2021)
8. Qian, X., Li, E., Zhang, J., Zhao, S.N., Wu, Q.E., Zhang, H., Wang, W., Wu, Y.: Hardness recognition of robotic forearm based on semi-supervised generative adversarial networks. Frontiers in Neurorobotics 13 (2019)
9. Schmitz, A., Bansho, Y., Noda, K., Iwata, H., Ogata, T., Sugano, S.: Tactile object recognition using deep learning and dropout. In: 2014 IEEE-RAS International Conference on Humanoid Robots. pp. 1044–1050 (2014)
10. Bottcher, W., Machado, P., Lama, N., McGinnity, T.: Object recognition for robotics from tactile time series data utilising different neural network architectures. In: 2021 International Joint Conference on Neural Networks (IJCNN). pp. 1–8 (2021)
11. Spiers, A.J., Liarokapis, M.V., Calli, B., Dollar, A.M.: Single-grasp object classification and feature extraction with simple robot hands and tactile sensors. IEEE Transactions on Haptics 9(2), 207–220 (2016)
12. Amin, Y., Gianoglio, C., Valle, M.: A novel tactile sensing system for robotic tactile perception of object properties. In: AISEM Annual Conference on Sensors and Microsystems. pp. 182–187. Springer (2023)
13. Jayakodi, N.K., Chatterjee, A., Choi, W., Doppa, J.R., Pande, P.P.: Trading-off accuracy and energy of deep inference on embedded systems: A co-design approach. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 37(11), 2881–2893 (2018)
14. Anguita, D., Ghio, A., Oneto, L., Ridella, S.: In-sample and out-of-sample model selection and error estimation for support vector machines. IEEE Transactions on Neural Networks and Learning Systems 23(9), 1390–1406 (2012)
15. Thant, H.A., San, K.M., Tun, K.M.L., Naing, T.T., Thein, N.: Mobile agents based load balancing method for parallel applications. In: 6th Asia-Pacific Symposium on Information and Telecommunication Technologies. pp. 77–82. IEEE (2005)
16. Noble, W.S.: What is a support vector machine? Nature biotechnology 24(12), 1565–1567 (2006)
17. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: theory and applications. Neurocomputing 70(1-3), 489–501 (2006)
18. Ragusa, E., Gianoglio, C., Gastaldo, P., Zunino, R.: A digital implementation of extreme learning machines for resource-constrained devices. IEEE Transactions on Circuits and Systems II: Express Briefs 65(8), 1104–1108 (2018)