Computationally Light Algorithms for Tactile Sensing Signals Elaboration and Classification

Youssef Amin, Christian Gianoglio, and Maurizio Valle
Department of Electrical, Electronic, Telecommunication Engineering and Naval Architecture DITEN
University of Genoa, Genova, Italy
Email: {youssef.amin, christian.gianoglio}@edu.unige.it, maurizio.valle@unige.it

***Abstract*—Tactile sensing systems require embedded processing to extract structured information in many application domains as prosthetics and robotics. In this regard, this paper proposes computationally light strategies to pre-process the sensor signals and extract features, feeding single layer feed-forward neural networks (SLFNNs) that proved good generalization performance keeping low the computational cost. We validate our proposal by integrating a tactile sensing system on a Baxter robot to collect and classify data from three objects of different stiffness. We compare different features extraction techniques and five SLFNNs to show the trade-off between generalization accuracy and computational cost of the whole processing unit. The results show that the processing unit that extracts the mean and standard deviation features from signals and adopts a fully connected neural network (FCNN) with 50 neurons and ReLu activation function achieves a high accuracy (94.4%) in the 3-class classification problem with a low computational cost, leading to the deployment on a resource-constrained device.**

***Index Terms*—Neural networks, Signal processing, Classification, Tactile sensing systems, Computational load, Tactile sensors, hardness recognition.**

I. Introduction

Tactile sensing systems are composed of tactile sensors and embedded processing units [1]–[3]. Tactile sensors translate mechanical stimuli such as bending, vibration, and grasp into electrical signals. A processing unit elaborates the signals in two stages: the first receives data from the acquisition system and performs pre-processing operations (e.g. filtering and features extraction), while the second infers useful information that can be interpreted by a user. Machine learning (ML) provides algorithms for this advanced processing, finding a correlation between the input signals and the output inference. These algorithms need to be computationally light because they must be deployed on embedded devices.

In this regard, this paper presents a study about the trade-off between generalization performance and the computational cost in terms of FLOPs of ML techniques based on single feed-forward neural networks (SLFNNs) to classify tactile signals. These networks proved a good balancing between classification accuracy and computational cost [4], [5]. Because the tactile signals need to be processed before feeding the predictors, we also adopt simple methods to filter data and extract features, evaluating their impact on the computational cost. To validate our study, we mounted a piezoelectric sensing patch on the gripper of a Baxter robot to collect data from the grasping of three objects, addressing a 3-class stiffness classification problem. An interface electronics (IE) sampled signals and applied a computational light moving average to filter out noise. A PC stored the filtered signals for further elaborations. Then, we applied a computationally light automatic technique to remove useless information from data and extract features. Eventually, we compared five SLFNNs that classified the stiffness of the objects based on the extracted features. The results show the capability of SLFNNs in achieving a good trade-off between generalization performance and computational cost measured as the number of FLOPs. These results urge to implement the whole processing unit on embedded devices to real-time elaborate tactile signals extracting high-level information. Overall, the main contributions of the paper are: 1) we propose computationally light pre-processing algorithms to filter noise out tactile signals and extract features; 2) we adopt computationally light SLFNNs to extract high-level information from tactile signals; 3) we compare the performance of the processing units both in terms of FLOPs and generalization accuracy on a use case problem, showing the possibility of implementing these units on embedded devices.

The remainder of the paper is: Section II presents the state of the art, Section III describes our proposal, Section IV depicts the experiments made on the use case, Section V presents the results, eventually Section VI concludes the paper.

II. State of the Art

In the last years, many tactile sensing systems have been presented. These systems include a sensing array followed by an elaboration unit. However, most systems are developed without a dedicated processing and acquisition unit. Indeed, some works adopted ML algorithms with a very high computational load, without keeping count of a possible embedded application. Some others addressed very specific tasks, making it troublesome to extend their proposals to a general-purpose application.

In [6], the authors employed a low cost e-skin and easy-designed PVDF piezoelectric polymer integrated into a flexible PCB to collected data from three touch modalities. Then, a PC classified the touch modalities with a tensor-based SVM algorithm, requiring a high computational effort. Recently, [7] adopted recurrent neural networks with a low number of parameters to address the predictor computational cost issues, achieving also a better classification accuracy. The

authors did not consider the computational load of the pre-processing stage. [8] presented a neuromorphic tactile sensing system for tactile-based texture recognition. The authors used a sensorized robot fingertip to perform sliding action over ten graded textures. A neuromorphic extreme learning machine (ELM) chip elaborated the input spikes and classified them into 10 classes. Although the authors highlighted the importance of energy consumption for real-life applications, they addressed a very specific task that is the texture recognition. [9] proposed a study targeting portable tactile sensing system, using a high resolution piezoresistive tactile sensing array. The authors acquired pressure images addressing a 22-class object recognition problem based on CNN algorithms. They also deployed the classifiers on different hardware platforms, comparing generalization performance, computational costs in terms of FLOPs, power consumptions, and inference times. The platforms included GPUs and/or powerful CPUs, making it troublesome to be employed as wearable devices. [10] developed a low cost scalable tactile sensing glove that contains an array of piezoresistive film connected by a conductive thread electrode. The authors collected a large scale tactile dataset of 135,000 frames that correspond to the interaction of the entire sensorized hand with 26 different objects and applied a deep convolutional neural network, namely the ResNet-18 architecture, in order to identify and explore typical tactile pattern of individual objects. Although the sensed glove was scalable and low cost, the elaboration unit required a huge computational cost. The authors in [11] installed the BioTAC fingertip tactile sensor to a motorized arm to generate tactile data by applying press and sliding actions on 14 materials divided into 6 groups. The materials were classified individually and according to their group type. The tactile data corresponded to signals from an array of 19 sensors and one thermistor (i.e., temperature sensor). The data were used to train seven different classifiers including hybrid structures that are implemented for material classification task. Moreover, principle component analysis (PCA) was applied to extract features and reduce the dimensionality of the inputs. As a result, the two stage support vector machine (SVM) approach had the best classification accuracy, also outperforming human by 16%. [12] also applied PCA for features extraction. Tactile data were collected from a WTS0406-38 tactile sensor integrated to a WSG 50 two-finger manipulator. The sensor array is composed of (4×6) tactile units converted to a vector sequence of 24 dimension time series data. The robot performed grasping operation on fruits and vegetables of different hardness in order to train and test the k-nearest neighbor (KNN) and SVM for hardness classification. The SVM achieved higher accuracy (94.27%). Both works [11] and [12] did not focus on the computational cost that is essential for the implementation of the proposed algorithms on embedded devices. [13] proposed a novel method based on a deep neural network model for hardness estimation, using data images from GelSight tactile sensor. The sensor was integrated on a robot fingertip to squeeze on objects of different shape and hardness. A sequence of images of the deforming gel

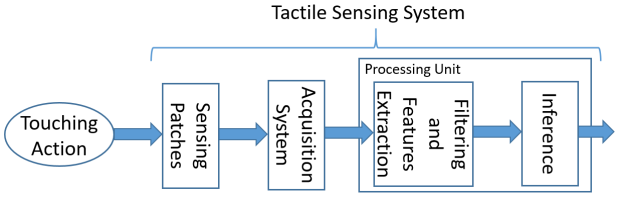

Fig. 1. Tactile system block diagram.

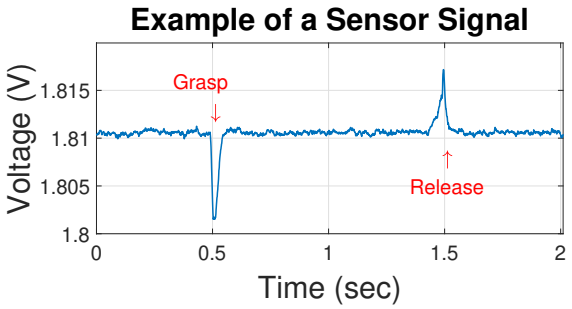

Fig. 2. Example of sensed signal by a P(VDF-TrFE) sensor.

were used to feed a convolutional neural network followed by a recurrent neural network. Although the model was able to estimate the hardness with an accuracy of 87% , but it was limited to hemispherical objects.

III. Proposal

A tactile sensing system is made of sensor patches and a processing unit. The sensor patches capture the touch signals. The processing unit elaborates the sensed signals in two main stages: the first stage is responsible for data acquisition, signals conditioning, and features extraction; the second stage makes predictions based on the extracted features. Fig. 1 shows the block diagram of the system. We propose computationally light techniques in terms of FLOPs in all the stages of the processing unit to elaborate tactile data and extract high-level information with high accuracy. In this way, the processing unit can be deployed on a resource-constrained device to be employed in a real-world application.

A. Sensor signals

In our tactile system, we utilized low-cost sensing patches based on the P(VDF-TrFE) (poly(vinylidene fluoride trifluoroethylene)) piezoelectric polymer sensors [14]. This type of sensor generates charges when subjected to external stimuli. Fig. 2 reports an example of a typical sensed signal. When contact occurs, the voltage across the sensor decreases then directly returns to its initial state while the sensor remains in contact with the object. Thus, it produces a drop peak (**Grasp** in the figure). On contrary, when the sensor is no more in contact with the object, the voltage increases and produces the **Release** peak.

B. Signal conditioning

An AD converter samples the sensed signals. To reduce the number of samples, we propose an automatic method based

on a thresholding mechanism that extracts only the samples concerning the grasp peak. The length n of the generated signal is defined by the designer and depends on the sampling frequency of the AD converter. Let call $\tilde{\boldsymbol{x}}$ the generated signal having n samples. To filter out noise from $\tilde{\boldsymbol{x}}$ we apply a two-samples moving average (MA): $y[i] = \alpha\tilde{x}[i] + (1-\alpha)y[i-1]$, where α is a user-defined parameter. According to [15] that reports the FLOP costs of the most frequently used operations, the MA filtering requires $3n$ FLOPs.

C. Features Extraction

To find a good representation of the data, we extract as features the statistical moments (i.e. the mean, standard deviation STD, skewness Sk, and kurtosis Ku) from the filtered signal $\boldsymbol{y}$ as they have a low impact on the computational load. Table I reports the FLOPs cost required by the features extraction. We considered that is possible to employ partial results of a feature to compute the others, saving FLOPs. As an example, the mean of a signal can be used to compute also the other statistical moments.

TABLE I
EXTRACTED FEATURES AND FLOPS COUNT

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

D. Inference Stage

We present five SLFNNs, that showed good compromise between generalization performance and computational cost, to output high-level information from tactile data. In general, the prediction function of a SLFNN is:

$$f(\boldsymbol{x}) = \sum_{n=1}^{N} \beta_n \phi(\boldsymbol{x} \cdot \boldsymbol{\omega}_n + b_n) \tag{1}$$

where $\boldsymbol{x}$ is the tested datum, N the number of neurons, $\boldsymbol{\beta}$ the weights between the hidden and output layers, ϕ an activation function, $\boldsymbol{\omega}$ the weights between the input and hidden layers, and $\boldsymbol{b}$ the biases. Three of the SLFNNs are based on the Extreme Learning Machine (ELM) paradigm [16] that bases on to the random extraction of the $\{\boldsymbol{\omega}, \boldsymbol{b}\}$ parameters, learning the $\boldsymbol{\beta}$ connections by solving a Regularized Least Square (RLS) problem. The first ELM network adopts the sigmoid activation function (we will refer to it as S-ELM); the second the ReLu activation function (R-ELM); the third the hard-limit activation function (H-ELM) and the $\boldsymbol{\omega}$ weights represented as a power of two (this simplification proved to be very effective for the deployment on low-cost and low-power devices [17], [18]). The other two SLFNNs are fully connected neural networks (FCNNs) trained through the back-propagation technique. The first employs the sigmoid activation function (S-FC), while the second the ReLu function (R-FC). In general, the networks based on the ELM paradigm require a training time much faster than FCNNs trained by the back-propagation, because they exploit the randomness to set the hidden parameters, learning only the last layer connections. Table II shows the FLOPs of the five SLFNNs with respect to the number of classes nc (i.e. the possible target labels in a classification problem, $nc = 1$ in case of regression), the number of neurons N, and the number of features nf of the input data.

TABLE II
SLFNNS AND FLOPS COUNT

SLFNNs	FLOPs Count
S-ELM	$N(2nf+14) + 2nc * N$
H-ELM	$N(2nf+1) + nc * N$
R-ELM	$N(2nf+2) + 2nc * N$
S-FC	$N(2nf+14) + 2nc * N$
R-FC	$N(2nf+2) + 2nc * N$

IV. USE CASE

To show the capability of our proposed processing unit in balancing the trade-off between generalization performance and computational cost, we integrated a P(VDF-TrFE) piezoelectric sensor patch on the gripper of the Baxter robot to collect tactile data performing grasp actions on three objects and classify their stiffness.

A. System setup

In the experiments, we customized the gripper of the Baxter robot to fit a sensing patch made of eight piezoelectric sensors. This patch possesses a total sensing area of $2.1 \times 1.1 cm^2$ distributed over 8 taxels (4×2 geometry) and a special resolution of center-to-center pitch of $1cm$. The taxel diameter is $2mm$. The patch provides a high-frequency bandwidth ranging from $1Hz$ to $1kHz$. In our experiments, we employed only four out of eight sensors because of the small dimensions of the manipulated objects. The integration procedure, presented in Fig. 3, involves adding a conductive tape, followed by a substrate and a protective layer on the bottom and the top side of the patch, to exclude environmental noises affecting the measurements. On the Baxter arm, we also integrated low-power interface electronics (IE) hosting an ARM-cortex M0 microcontroller. The IE acquires simultaneously the tactile signals from four channels that correspond to the four sensors, at $2kSps$ sampling frequency. The IE filters the data by the moving average technique presented in Section III-B, and continuously transmits the filtered signals to the PC using USB communication. We developed a graphical user interface (GUI) on LabVIEW to control the data collection process. The PC hosted the automatic detection of the grasp peaks (Fig. 2), the features extraction, and the inference algorithms, as illustrated in Section III.

B. Datasets Collection

Fig. 4 shows the three objects employed during the experimental session. They have a cylindrical shape and are made of different materials that characterize the stiffness: 1) a deformable rubber-like object with slight stiffness (SR), 2) a harder rubber-like object (HR), and 3) a rigid object

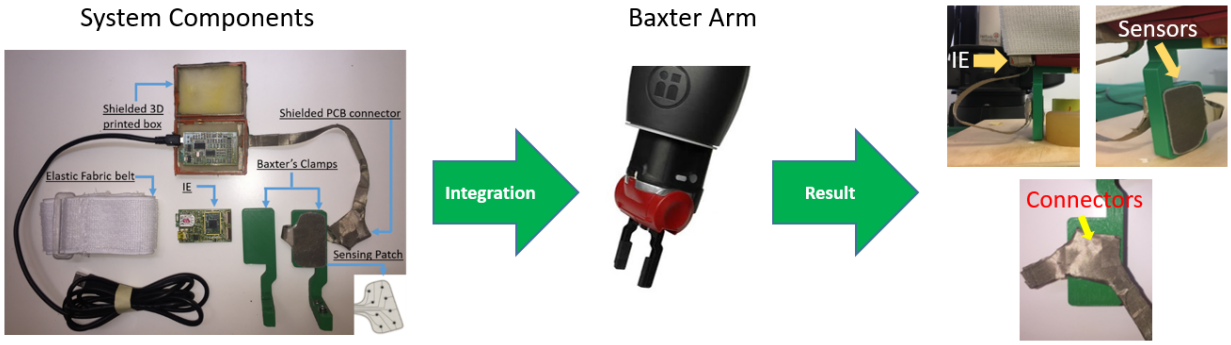

Fig. 3. Integration of the tactile sensing system on the Baxter robot arm

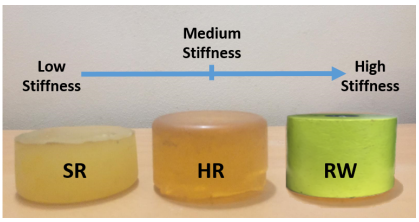

Fig. 4. Objects of different stiffness: Soft Rubber (SR), Hard Rubber (HR), and Rigid Wood (RW).

made of wood (RW). We collected data by performing 20 grasp/release sequential trials on each object 12 times, modifying the position of Baxter arm each time, paying attention that at least one sensor out of the four was in contact with the manipulated object. Eventually, we obtained 240 grasping operations on each object. Each trial takes 3 seconds: 1 second of grasping and 2 seconds of release. During experiments, we set the velocity and force of the gripper to $v = 5cm/s$ and $F = 0.03 * Fmax$ ($Fmax = 35N$), to avoid any damage for the sensors. The IE continuously acquired the tactile data from four sensors simultaneously, and transmitted them to the PC for further elaboration. Fig. 5 illustrates the processing of the tactile data until the features extraction. Following the procedure described in Section III, we automatically extracted the grasp peaks (samples between dashed lines in Fig. 5) by taking 30 samples before the minimum peak value and 120 after. The reducing operation is triggered by the overcoming of a threshold on at least one channel. Consequently, the samples extraction is applied on all the channels as Fig. 5 shows. After, we computed the statistical moments, described in Section III-C, from the generated signals (i.e., 4 features for each channel). As a result, we obtained 720 data (240 for each object) represented by 16 features. For the sake of comparison, we built three 3-class datasets: the first one (4F) contains the 720 data with 4 features (i.e. the average from each channel), the second (8F) includes 720 data with 8 features (i.e. average and standard deviation), the third (16F) consists of 720 data with all the 16 features. We normalized each feature of all the datasets having zero mean and standard deviation equals 1.

C. Training Strategy

All the five classifiers presented in Section III-D were designed to deal with a 3-class classification problem, thus the output layers present 3 neurons, one for each class. The Softmax function assigns the label (i.e. SR, HR, and RW) to the neuron with the highest probability. We employed a grid search in the model selection phase to fix the algorithms architectures. The pool of parameters candidates was:

- $N = \{25, 50, 100, 200, 300\}$ number of hidden neurons;
- $N_{max} = \{50, 100, 200, 300\}$ maximum number of hidden neurons;
- $\lambda = \{10^i,\ i = -8, -3, ..., 8\}$ L2 regularizer.

Moreover, the hidden parameters $\{\boldsymbol{\omega}, \boldsymbol{b}\}$ for ELM networks were extracted between $(-1,\ +1)$, and the weights for FCNN were initialized in the same interval. Each experiment involved 10 extractions of training/validation/test sets, in which 70% of data were used as training and the remaining 30% equally split in validation (to choose the best models) and test (to estimate the generalization performance). Only in the ELM networks, we also extracted the random parameters 10 times for each training/validation/test trial. We designed all the models in python, using the Keras library for the FCNNs. In the latter case, we set the number of epochs to 200, the patience on the validation accuracy to 20 to implement an early stopping criterion, the batch size to 64, and the learning rate to 0.01. During the learning phase, we constrained the models to be trained between the minimum number of neurons (i.e. $N = 25$) and the current maximum, set by N_{max}. For example, when $N_{max} = 100$ we trained each classifier with $N = 25, 50, 100$, looking for the architecture that achieves the best generalization performance on the validation set.

V. Results

In this section, we evaluate our proposed processing unit in balancing the trade-off between generalization performance and computational cost. We tested the five SLFNNs with the three datasets described in Section IV-B, showing the FLOPs required by the stages to elaborate the data.

Concerning the FC-based networks, we computed the generalization performance on the test set by averaging the results of the 10 splits in training/validation/test sets. Instead, for the

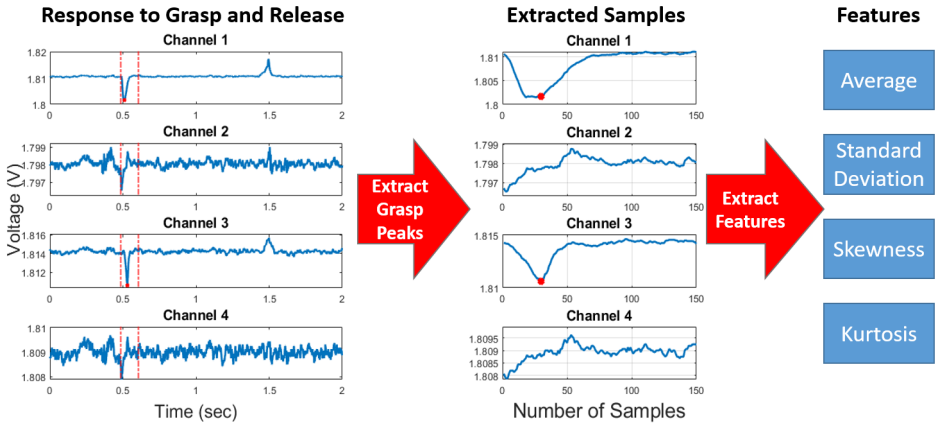

Fig. 5. Data pre-processing and features extraction.

ELM-based networks, we averaged the results of 100 trials, i.e. the 10 splits and 10 random extractions of the hidden parameters $\{\boldsymbol{\omega}, \boldsymbol{b}\}$ for each split.

Regarding the computational load, we calculated the FLOPs of the whole processing unit, as detailed in Section III. During the signal conditioning, we considered the cost of the moving average only applied at the grasp peaks with $n = 150$ samples. We adopted this simplification because in the real application the IE will wait for a trigger (by setting a threshold) before filtering and storing each grasp peak. Similarly to the generalization performance, we averaged the number of neurons to estimate the classifiers FLOPs according to Table II.

Fig. 6 shows the averaged generalization performance on the test set of each inference model, with $N_{max} = 100$, in the function of the computational load of the whole processing unit represented as the number of FLOPs, for all the three datasets (i.e., 4F, 8F, and 16F). The processing units that present the lowest computational load are based on the 4F dataset. The R-ELM achieves the best generalization performance (i.e. 87.2%) classifying this dataset. Increasing the number of extracted features, the FC outperforms the ELM networks requiring also a lower number of FLOPs. The best accuracies, 94.7% and 95.4% respectively, are obtained by R-FC with both the 8F and 16F datasets. In general, the accuracy-FLOPs trends show that by employing the 8F dataset we obtained a big improvement in the generalization performance (up to 15%) despite a higher computational cost with respect to the 4F dataset; on the other hand, adopting the 16F dataset, the accuracies are almost the same or even worse in case of S-ELM and R-ELM with respect to the 8F dataset.

Table III reports the results about the average generalization performance and FLOPs of the whole processing for all the configurations of N_{max} of the five SLFNNs. The first column represents the five classifiers, the second column the configurations of N_{max}, the columns from the third to the fifth the generalization performance obtained classifying the 3 datasets, while the last three columns the FLOPs required by the processing units in elaborating and classifying the 3

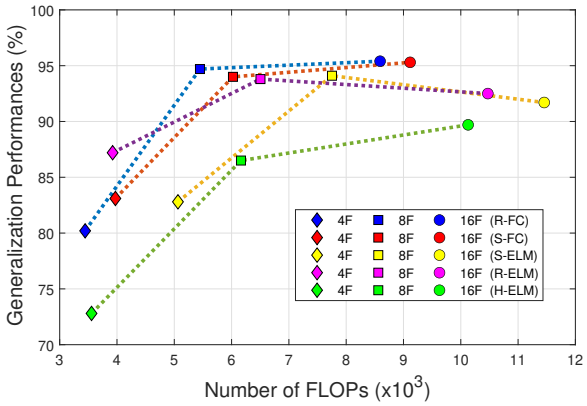

Fig. 6. Generalization Performance vs Computational Load with $N_{max} = 100$.

datasets. For each classifier, the table sets as reference the results obtained with $N_{max} = 100$, and sketches for each dataset the differences between the references and the other configurations: concerning the generalization performance a positive green number means an improvement, while a negative red represents a drop; on the contrary, as regards the FLOPS, a negative green number indicates a lower computational load, and red positive one a greater consumption. The results show that the FC networks generalization performance for each dataset does not or slightly changes by modifying the number of neurons. Thus, with $N_{max} = 50$, the FC classifiers present a good accuracy with the lowest computational cost with respect to the other solutions. Conversely, with the ELM networks, when increasing the number of neurons, we obtain higher accuracy (the best is 96.3% reached by S-ELM with 8F and N_{max}=300), but also with higher computational costs (with $N_{max} = 300$ the cost almost doubles that obtained with $N_{max} = 100$ configuration). The designer, during the deployment of the processing unit on a resources-constrained device, has to keep count of the trade-off between generalization performance and computational cost. Nevertheless, the

number of FLOPs is low in general for all the processing units, allowing to implementation of the best solution in terms of generalization performance on an embedded device.

TABLE III
SLFNNs GENERALIZATION PERFORMANCE AND PROCESSING UNIT FLOPS

Algorithms		Av. Accuracy (%)			FLOPs		
		4F	8F	16F	4F	8F	16F
R-FC	50	-0.4	-0.3	+0.1	-320	-408	-560
	100	**80.2**	**94.7**	**95.4**	**3,444**	**5,448**	**8,592**
	200	+1.2	+0.1	0	+576	+840	+400
	300	+1.5	0	+0.1	+1,616	+1,176	+1,080
S-FC	50	+0.1	0	0	-532	-432	-676
	100	**83.1**	**94.0**	**95.3**	**3,972**	**6,024**	**9,116**
	200	-0.2	0	+0.2	+532	+648	+884
	300	0	+0.1	+0.1	+1,260	+1,224	+2,496
R-ELM	50	-3.5	-4.9	-4.7	-720	-1,080	-1,800
	100	**87.2**	**93.8**	**92.5**	**3,924**	**6,504**	**10,472**
	200	+2.4	+1.5	+1.6	+1,424	+1,992	+3,360
	300	+3.4	+1.9	+2.1	+2,336	+3,024	+5,520
S-ELM	50	-3.4	-3.9	-4.2	-1,288	-1,728	-2,236
	100	**82.8**	**94.1**	**91.7**	**5,064**	**7,752**	**11,456**
	200	+1.6	+1.8	+2.1	+2,044	+3,024	+4,108
	300	+1.9	+2.2	+2.8	+3,388	+5,148	+6,864
H-ELM	50	-9.0	-6.7	-5.6	-588	-960	-1,692
	100	**72.8**	**86.5**	**89.7**	**3,556**	**6,164**	**10,128**
	200	+7.0	+3.3	+3.1	+1,212	+1,860	+3,384
	300	+9.9	+4.6	+3.8	+2,196	+3,340	+6,156

VI. CONCLUSION

In this paper, we proposed a two stages processing unit for the elaboration and classification of tactile signals, balancing the generalization performance and the computational cost in terms of FLOPs. We collected data by integrating a tactile sensing system on a Baxter robot to address a 3-class classification problem concerning the inference of the stiffness of three manipulated objects. We employed a moving average to filter the data, then we extracted four features from each channel based on the statistical moments. The features fed 5 single-layer feed-forward neural networks to infer the stiffness. The results showed that the processing unit based on the fully connected network with $N_{max} = 50$ and ReLu activation function achieved the best computational cost considering all the three datasets (4F, 8F, and 16F), and a high generalization performance (79.8%, 94.4%, and 95.5%, respectively). However, the extreme learning machine network with sigmoid activation function presented the highest classification accuracy (96.3%) with the 8F dataset but with a higher computational cost. In general, our processing unit showed a good trade-off between classification accuracy and computational cost, leading also to a possible deployment on resource-constrained devices.

ACKNOWLEDGEMENT

The authors acknowledge partial 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] C. Bartolozzi, L. Natale, F. Nori, and G. Metta, "Robots with a sense of touch," *Nature materials*, vol. 15, no. 9, pp. 921–925, 2016.

[2] R. S. Dahiya, P. Mittendorfer, M. Valle, G. Cheng, and V. J. Lumelsky, "Directions toward effective utilization of tactile skin: A review," *IEEE Sensors Journal*, vol. 13, no. 11, pp. 4121–4138, 2013.

[3] M. M. Iskarous and N. V. Thakor, "E-skins: Biomimetic sensing and encoding for upper limb prostheses," *Proceedings of the IEEE*, vol. 107, no. 10, pp. 2052–2064, 2019.

[4] W. Cao, X. Wang, Z. Ming, and J. Gao, "A review on neural networks with random weights," *Neurocomputing*, vol. 275, pp. 278–287, 2018.

[5] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018.

[6] P. Gastaldo, L. Pinna, L. Seminara, M. Valle, and R. Zunino, "Computational intelligence techniques for tactile sensing systems," *Sensors*, vol. 14, no. 6, pp. 10 952–10 976, 2014. [Online]. Available: https://www.mdpi.com/1424-8220/14/6/10952

[7] M. Alameh, Y. Abbass, A. Ibrahim, G. Moser, and M. Valle, "Touch modality classification using recurrent neural networks," *IEEE Sensors Journal*, 2021.

[8] M. Rasouli, Y. Chen, A. Basu, S. L. Kukreja, and N. V. Thakor, "An extreme learning machine-based neuromorphic tactile sensing system for texture recognition," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 2, pp. 313–325, 2018.

[9] M. Alameh, Y. Abbass, A. Ibrahim, and M. Valle, "Smart tactile sensing systems based on embedded cnn implementations," *Micromachines*, vol. 11, no. 1, 2020. [Online]. Available: https://www.mdpi.com/2072-666X/11/1/103

[10] S. Sundaram, P. Kellnhofer, Y. Li, J.-Y. Zhu, A. Torralba, and W. Matusik, "Learning the signatures of the human grasp using a scalable tactile glove," *Nature*, vol. 569, no. 7758, pp. 698–702, 2019.

[11] E. Kerr, T. McGinnity, and S. Coleman, "Material recognition using tactile sensing," *Expert Systems with Applications*, vol. 94, pp. 94–111, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417417307273

[12] Z. Zhang, J. Zhou, Z. Yan, K. Wang, J. Mao, and Z. Jiang, "Hardness recognition of fruits and vegetables based on tactile array information of manipulator," *Computers and Electronics in Agriculture*, vol. 181, p. 105959, feb 2021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0168169920331641

[13] W. Yuan, C. Zhu, A. Owens, M. A. Srinivasan, and E. H. Adelson, "Shape-independent hardness estimation using deep learning and a gelsight tactile sensor," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 951–958.

[14] H. Fares, Y. Abbass, M. Valle, and L. Seminara, "Validation of Screen-Printed Electronic Skin Based on Piezoelectric Polymer Sensors," *Sensors*, vol. 20, no. 4, p. 1160, feb 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/4/1160

[15] H. A. Thant, K. M. San, K. M. L. Tun, T. T. Naing, and N. Thein, "Mobile agents based load balancing method for parallel applications," in *6th Asia-Pacific Symposium on Information and Telecommunication Technologies*. IEEE, 2005, pp. 77–82.

[16] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[17] E. Ragusa, C. Gianoglio, P. Gastaldo, and R. Zunino, "A digital implementation of extreme learning machines for resource-constrained devices," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 8, pp. 1104–1108, 2018.

[18] E. Ragusa, C. Gianoglio, R. Zunino, and P. Gastaldo, "A design strategy for the efficient implementation of random basis neural networks on resource-constrained devices," *Neural Processing Letters*, pp. 1–19, 2019.