1-D Convolutional Neural Networks for Touch Modalities Classification

Christian Gianoglio, Edoardo Ragusa, Rodolfo Zunino and Maurizio Valle
Department of Electrical, Electronic, Telecommunication Engineering and Naval Architecture DITEN
University of Genoa, Genova, Italy
Email: {christian.gianoglio, edoardo.ragusa}@edu.unige.it, {rodolfo.zunino, maurizio.valle}@unige.it

2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS) | 978-1-7281-8281-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICECS53924.2021.9665576

***Abstract*—Artificial tactile systems can facilitate the life of people suffering from a loss of the sense of touch. These systems use sensors and digital, battery-operated embedded units for data processing. Therefore, low-power, resource-constrained devices should host those embedded devices. The paper presents a framework based on 1-D convolutional neural networks (CNNs), which tackles the problem of classifying touch modalities, while limiting the number of architecture parameters. The paper also considers the computational cost of the pre-processing stage that handles tactile-sensor data before classification. The related pre-processing unit affects resources occupancy, computational cost, and ultimately classification accuracy. The experimental session involved a state-of-the-art real-world dataset containing three touch modalities. The 1-D CNN outperformed existing solutions in terms of accuracy, and showed a satisfactory trade-off between accuracy, computational cost, and resources occupancy. The implementation of the 1-D CNN classifier on an Arduino Nano 33 BLE device yielded real-time performances.**

***Index Terms*—Touch modalities classification, CNNs, embedded systems.**

I. Introduction

In the last years, tactile sensing systems were used for assisting humans who lost the sense of touch and helping them interact with the surrounding environment. An effective tactile sensing system should include tactile sensors and processing units, which should handle information from sensors in real-time to provide valuable feedback to users. These systems should be deployed on resource-constrained devices, especially when targeting use in daily life. Thus balancing power consumption, size, and latency becomes a considerable challenge.

The paper tackles the classification of a set of touch modalities when applied to an electronic skin. Toward that goal, the system should correlate in real-time the activations of several sensors placed over the skin to infer the overall action. Artificial Intelligence (AI) models can support the effective and efficient elaboration of the sensed signals. From among the many existing AI techniques, one may summarize that "traditional" machine learning relies on hand-crafted features to support inference. By contrast, deep-learning methods extract features directly from data; this ability comes at the expense of a large number of parameters, thus inflating memory requirements, computational load, and energy consumption. Indeed, traditional implementation approaches seem more suitable for highly constrained scenarios [1], [2]. On the other hand, the gap in accuracy performances is wide enough that most of the research activity is shifting progressively towards deep learning. The deployment of deep networks on low-power devices still proves a crucial issue due to the large number of parameters that characterize these models. In fact, deep networks with a small number of parameters might easily satisfy tight hardware constraints, and therefore be hosted on low-power, inexpensive commercial micro-controllers for real-time applications.

This paper presents an end-to-end framework based on 1-D convolutional neural networks (CNNs) to address the classification of touch modalities, while limiting the computational cost (in terms of FLOPs) and the resource occupancy (in terms of the number of parameters). A matrix of sensors acquires the signals which are characterized by a 3-D tensor structure, in which two dimensions represent the geometry of the sensors patch, whereas time is the third dimension. The pre-processing unit that reduces noise and data dimensionality before feeding the CNN ultimately affects the resources occupancy, power consumption, and classification accuracy.

The main contributions of the presented research can be summarized as follows:

- adopting 1-D CNNs to classify touch modalities can outperform existing solutions in terms of accuracy;
- the 1-D CNN approach also compares favourably with state-of-the-art solutions on the same problem, when considering the computational cost (FLOPs) and the number of parameters;
- deploying the 1-D CNN classifier on an low-cost Arduino Nano 33 BLE micro-controller still yielded real-time performances.

In the following, Section II overviews related works in this area of research, while Section III describes the proposed approach; Section IV illustrates the experimental set-up, whereas Section V reports on the obtained results. Concluding remarks are made in Section VI.

II. Related Works

Prosthetics and robotics widely adopt tactile sensing systems, either to restore the loss of sense of touch in humans or to augment robot manipulation by providing feedback to human operators.

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.

A decision-tree predictor [3] classified four categories of touch-patterns, which were collected by 3x3 sensors arrays and integrated with accelerometer information. In [4], [5], the authors employed the LogitBoost algorithm [6] to discriminate eights types of touch modalities. The approach described in [7] involved a set of biologically inspired features and Support Vector Machines (SVMs) to categorize nine touch modalities, which were collected on a humanoid covered with an artificial skin. Those results outperformed the accuracy values obtained with the features collected by [5]. In [8], the authors compared different algorithms to recognize 14 social touch gestures, and reported that a kernel SVM attained the best accuracy. Similar accuracy scores on the same dataset were reported in [9], when applying deep-learning algorithms. An image-based deep network [10] classified 6 touch modalities, which had been acquired by a camera under a translucent robot skin while interacting with a non-rigid robot.

The approach described in [11] handled a dataset covering three touch modalities, collected by piezoelectric sensors and represented as tensor data; when applying tensor-SVM and tensor-RLS algorithms, the 3-category classifiers attained average generalization accuracies of 71% and 73.7%, respectively. Several subsequent works applied different techniques on the same dataset to improve generalization performance and reduce computational costs. In [12], [13], the authors adopted the k-NN and the SVM models to address a two-class classification problem; approximate computing techniques reduced the execution time and memory usage during the inference phase.

A different approach [14] transformed the tensor data into RGB images, and involved deep neural networks trained with transfer learning techniques. Adopting an Inception Resnet model resulted in a notable classification accuracy (76.9%) but implied an excessive computational cost. In [15], the authors applied recurrent neural networks, to take into account timing dependencies within samples and easily deal with 3D tensor data. The three-modality classifiers exhibited good results in terms of average accuracy (74.5%) and a sharp reduction in the predictors cost. Principal component analysis was also applied to reduce the dimensionality of the feature space [16], handled by a singular value decomposition-based kernel, and a global alignment kernel to map the data into a suitable space supporting kernel-SVM classification. This method scored an average generalization performance of 85.4 %, at the expense of an impractical computational cost for data processing.

III. 1-D CNNs FOR TOUCH-MODALITY CLASSIFICATION

The real-time classification of touch modalities requires that sensor data be processed before feeding the predictor. The approach proposed in this paper, therefore, involves a two-stage system: in the first stage, data from the acquisition system undergo a set of pre-processing operations, which feed the second stage, where the actual inference of the touch modality is worked out. Figure 1 outlines the overall processing system.

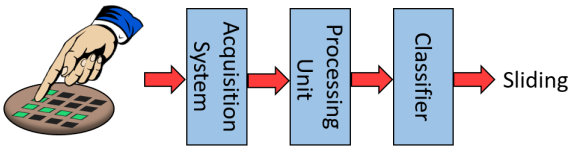

Fig. 1. Elaboration system.

A. Data structure

The data acquired by the tactile sensor matrix form 3-dimensional tensors; two dimensions relate to the geometry of the sensor matrix, whereas the third dimension conveys time information. Figure 2 illustrates the nature of the input tensor.

B. Pre-processing

Raw sensor signals are noisy and mostly contain useless information that needs to be filtered out. In principle, CNNs could filter out implicitly useless contents from input data; that filtering capability, however, would increase the number of parameters of the network and therefore hinder low-resource implementations. To limit the computational cost, a straightforward moving-average algorithm (outlined in Procedure 1) handles signal data and applies a moving window covering 50% of overlapped samples. This mechanism reduces both noise and the number of signals samples. The resulting 3-D tensors feed the second stage, which contains the CNN and extracts features from complex data structures.

Procedure 1 Moving Average with 50% of overlapping samples

Input

- Dataset $\mathcal{D} = \{\mathcal{X}_i \in \mathbb{R}^{M \times N \times T}; i = 1, ..., N_D\}$
- Desired number of slices Ns

1. **Methodology**

$wL = 2 * length(\mathcal{X}_1(1, 1, :))/(Ns + 1)$
for $(i = 1; i \leq N_D; i++)$ **do**
  for $(m = 1; m \leq M; m++)$ **do**
    for $(n = 1; n \leq N; n++)$ **do**
      for $(s = 1; s \leq Ns; s++)$ **do**
        if $s == 1$ **then**
          $\tilde{\mathcal{X}}_i(m, n, s)$=mean$(\mathcal{X}_i(m, n, 1 : wL))$
          $wL = wL/2$
        else
          $\tilde{\mathcal{X}}_i(m, n, s)$=mean$(\mathcal{X}_i(m, n, wL * (s - 1) + 1 : wL + s * wL))$
        end if
      end for
    end for
  end for
end for

2. **Output** Return $\mathcal{D}_{Ns} = \{\tilde{\mathcal{X}}_i \in \mathbb{R}^{M \times N \times Ns}; i = 1, ..., 840\}$

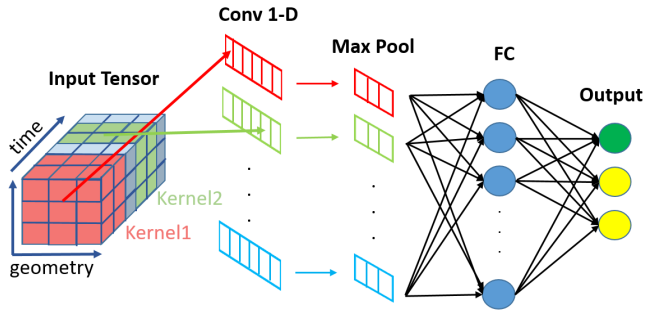

Fig. 2. 1-D CNN example.

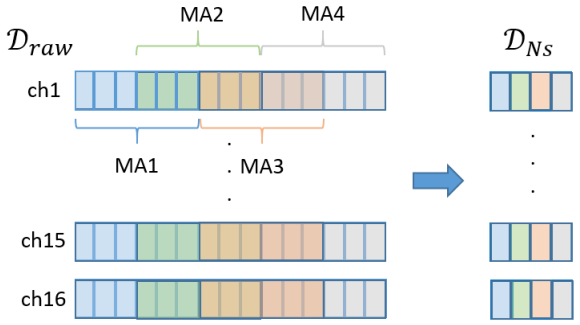

Fig. 3. Moving average with 50% of overlapped samples applied to $\mathcal{D}_{Raw}$.

C. Classifier

A 1-D CNN classifies the pre-processing results worked out in the first stage. As compared with 2-D CNNs that mostly address images and video applications, 1-D CNNs can process 1-D time sequences. In fact, these networks can handle input tensors featuring several dimensions, but the convolution operation always involves one dimension only. The method, therefore, takes advantage of the fact that touch-modality data are temporal signals with a 3-D tensor structure, and ultimately exploits the capability of 1-D CNNs to automatically learn patterns from temporal sequences. As a major advantage, a real-time, low-cost hardware implementation is feasible thanks to the simple, compact nature of 1-D convolutions, just involving scalar multiplications and additions [17].

Figure 2 gives an example of a 1-D CNN, embedding one convolutional layer (**Conv 1-D**) that processes a 3-D tensor: two dimensions correspond to the geometry of sensors, while the third dimension gives time information. The convolution consists of f 3-D kernels, having the same geometry of the input tensor but with designer-fixed third dimension, that slide along the time direction, creating f vectors (features) through the scalar product operation. In this way, the CNN can learn both the spatial and the temporal dependencies within the signals of the input tensors [17]: at each step along the time axis, the kernels merge geometrical information and yield one scalar value, as shown in Fig. 2 (Kernel1 and Kernel2 work out one value for each step along the time axis). Activation functions such as ReLu or sigmoid usually introduce some non-linearity after the convolutional layer. Subsequently, pooling techniques (denoted as **Max Pool** in the Figure) reduce the dimensions of the f features and aggregate local information, for example, by extracting the maximum value from non-overlapping patches. Finally, a fully connected layer (marked as **FC** in the Figure) including non-linear activation functions gets feature values from the topmost convolutional/pooling and connects to the output layer. The number of neurons in the latter layer corresponds to the number of possible output categories, and the output neuron with the highest activation marks the predicted class.

When envisioning deep architectures with several convolutional layers, the features extracted by a lower-level layer just provide the inputs to the subsequent convolution layer.

IV. Experimental Setup

A. Dataset

Seventy subjects performed three touch modalities [11], namely, "slide a finger", "brush a paintbrush", and "rolling a washer"; each touch action was repeated two times on a 4x4 piezoelectric sensors matrix in two directions starting from a random position. The resulting dataset held 840 tensor data, 280 per class: each datum was a 3-modes tensor, in which the first two modes represented the 4x4 sensors geometry (i.e. 16 channels), whereas the third dimension carried time information (each action, sampled at 3KSamples/s frequency, lasted 10 seconds). In the following, the dataset is formalized as: $\mathcal{D}_{Raw} = \{(\mathcal{X}, y)_i; \mathcal{X}_i \in \mathbb{R}^{4\times4\times30000}; y \in \{Slide, Brush, Roll\}; i = 1, ..., 840\}$.

B. Pre-processing Techniques

Raw sensor signal underwent the moving average procedure 1 (overlap window covering 50% of samples) to reduce the tensor dimensions. The moving average applied along the time axis in $\mathcal{D}_{Raw}$, and adopted four-time windows amplitudes. This resulted in four datasets $\mathcal{D}_{Ns}$, where $Ns = \{30, 60, 100, 1000\}$ was the number of slices of the extracted tensors.

To illustrate the pre-processing Procedure 1, Figure 3 gives an example of the moving average (MA) when applied to the 16 channels of $\mathcal{D}_{Raw}$. In the Figure, the $\mathcal{D}_{Raw}$ channels contains 15 samples and $Ns = 4$.

For the sake of comparison, an alternative pre-processing step adopted the strategy proposed in [15] to filter the original dataset $\mathcal{D}_{Raw}$: before applying the moving average, a thresholding mechanism reduced the number of samples in each data channel from $n = 30000$ to $\hat{n} = 6000$, thus creating the $\mathcal{D}_{Thr}$ dataset. As a result, four datasets $\hat{\mathcal{D}}_{Ns}$ originated from $\mathcal{D}_{Thr}$, holding the same number of time slices of $\mathcal{D}_{Ns}$. All the datasets had the same structure of $\mathcal{D}_{Raw}$, but with a different number of time samples accordingly to the number of slices. The pre-processing phase was completed by normalizing all the datasets to zero mean and standard deviation equals 1.

C. Training Strategy

The training procedure involved model selection, i.e. the tuning of the classifier architecture hyper-parameters. That

procedure explored a grid of candidates, privileging solutions with a low number of parameters and a low computational cost:

- number of convolutional layers from 2 to 3 (the filter candidates were: $(4, 4)$, $(4, 8)$, $(8, 8)$, $(4, 4, 4)$, $(8, 8, 8)$, $(4, 8, 16)$);
- kernel size $Ks = \{4, 8, 10\}$.

The filters represented the number of kernels applied to the input features in each convolutional layer. For example, setting $f = (4, 8, 16)$ implied the use of three convolutional layers, the application of 4 kernels to the input tensor, and the doubling in the number of kernels at each next convolutional layer. For each hyper-parameter setting, the experiments trained 18 CNN models, thus covering all possible combinations of filters and kernel sizes, on each dataset. The ReLu activation function characterized every convolutional layer, followed by a max-pooling layer to halve the number of samples between two consecutive convolutions. To reduce the models number of parameters, a convolutional layer replaced fully connected layers, followed by a non-linearity, that included three output filters (i.e., as many as the number of touch-modality classes). The max-pooling layer reduced the dimensions of each filter to 1. Eventually, a softmax function prompted the predicted label associated with the input signal.

To ensure a fair comparison with state-of-the-art solutions, the experiments included the implementation of the LSTM network for the same classification problem [15], as it showed a notable accuracy and featured a good trade-off between accuracy and computational cost. A pool of neurons ($N = \{10, 25, 50, 100\}$) included the explored options to pinpoint the best configuration during model selection.

In the experiments, the predictors were trained on all the eight datasets $\mathcal{D}_{Ns}$ and $\hat{\mathcal{D}}_{Ns}$. Each test involved 10 runs, that is, 10 random training/validation/test sets. In each run, 70% of data were used for training, whereas the remaining 30% were equally split into a validation set (for model selection) and a test set (to estimate generalization performance). The models were all implemented in Python by using the Keras API, with the following settings: number of epochs = 200, the patience on the validation accuracy = 20 (to support early stopping), batch size = 40, and learning rate = 0.001.

D. Embedded device

To test the time performance during the inference phase, we deployed the 1-D CNN on a Arduino Nano 33 BLE, as it could be easily embedded in a prosthetic tactile system. That low cost and resources-constrained device featured a CPU clock frequency = $64MHz$, a flash memory size = $1MB$, SRAM holding $256kB$, and operated at $3.3V$. Upon training completion of the 1-D CNN, the Python Tensorflow library allowed to export the float32 and int8 models in the tflite format. Deploying both those models on the device made it possible to test the effects of each quantized model on inference timings and the possible losses in generalization. The inference-timing measurements averaged results on 120 input data (i.e. the test set dimension), also measuring, at the same time, the increment in miss-classified data when adopting the int8 model as opposed to the original float32 one.

V. Experimental Results

This section presents the generalization performances obtained by the deep networks on the tested datasets, the associate computational costs in terms of FLOPs, the number of parameters of the processing systems (including both pre-processing and classification), and the results attained when deploying the architectures on the Arduino Nano 33 BLE.

A. Generalization performance Results

Figure 4 reports on the generalization performances (measured on the test set) attained by the compared deep networks (i.e. the proposed 1-D CNN and LSTM [15]) on the various datasets. In the x-axis of each graph, the four leftmost items refer to the pre-processing techniques adopted in [15] that generated $\hat{\mathcal{D}}_{Ns}$ datasets; the four rightmost items correspond to the preprocessing approach proposed in Section IV.B ($\mathcal{D}_{Ns}$), i.e., when applying a moving average on the raw data $\mathcal{D}_{Raw}$. The figure gives the boxplots of the measured accuracies (on the test set) computed on each dataset over the 10 test sets (i.e. 10 runs). The red lines highlight the median values within each boxplot, the blue box delimits the first and the third quartiles, while the whiskers mark the variability outside the upper and lower quartiles.

The reported results show that the 1-D CNN always outperformed (in terms of the median value) the LSTM on each dataset, except with $\hat{\mathcal{D}}_{30}$ and $\mathcal{D}_{30}$; in the latter cases, anyway, the differences proved marginal (75% and 80.8% for the CNN, as opposed to 78.3% and 81.2%, respectively, for LSTM). The variabilities featured by the 1-D CNN were lower or most similar to the LSTM ones, and indicated a tighter spread over the different runs. The experiments also pointed out that a larger number of time slices in the LSTM corresponded to higher variability in accuracy, also coming with lower median values.

In the 1-D CNN, the highest accuracies were obtained on the $\mathcal{D}_{Ns}$ datasets, and the best median (85.0%) was achieved on $\mathcal{D}_{100}$. The best-performing network included 3 convolutional layers, with a filter dimension $f = 8$ and a kernel size $ks = 10$ for each convolutional layer.

B. FLOPs and Parameters Results

Resource occupancy comparisons for both network models and for each dataset measured the FLOPs and the number of parameters in each best-performing architecture. The evaluation of the FLOPs also kept into account the pre-processing operations, since they ultimately affected the resources of the target implementation devices. Table I reports the measured Mega-FLOPs (MFLOPs) and the number of parameters of the two networks for each dataset.

The table shows that, for each dataset, the LSTM always required a larger number of FLOPs and parameters than those required by the CNN, with the exceptions of $\mathcal{D}_{100}$ and $\mathcal{D}_{1000}$. In the latter cases, however, LSTM scored much lower median

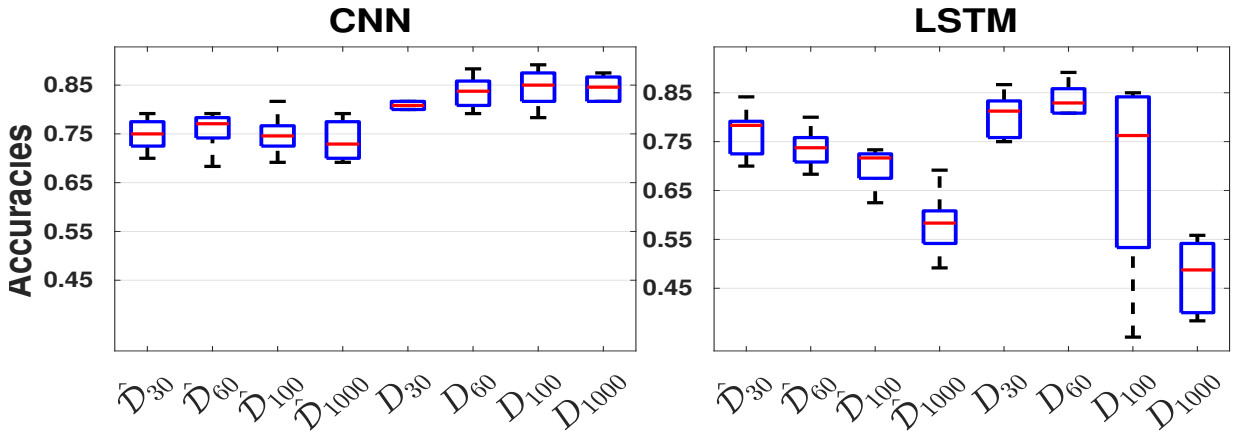

Fig. 4. Boxplots of the accuracies obtained by the deep networks on the eight datasets, computed on the test sets over 10 runs.

TABLE I
DEEP NETWORKS NUMBER OF FLOPS AND PARAMETERS

Dataset	CNN		LSTM	
	MFLOPs	Params	MFLOPs	Params
$\hat{\mathcal{D}}_{30}$	0.28	1579	1.12	13553
$\hat{\mathcal{D}}_{60}$	0.38	2099	2.04	13553
$\hat{\mathcal{D}}_{100}$	0.53	1963	1.23	4278
$\hat{\mathcal{D}}_{1000}$	3.46	1963	30.77	13553
$\mathcal{D}_{30}$	1.03	2611	1.85	13553
$\mathcal{D}_{60}$	1.12	2099	2.78	13553
$\mathcal{D}_{100}$	1.31	2611	1.24	1113
$\mathcal{D}_{1000}$	4.53	2611	3.89	1113

accuracies, as per Figure 4. So one may conclude that the 1-D CNN outperformed the LSTM in terms of accuracy, also involving a lower computational effort and a smaller amount of parameters to be stored in the target implementation device. In the comparison among the various 1-D CNN networks, the $\mathcal{D}_{Ns}$ datasets required a larger number of FLOPS (about three-time with $Ns = \{30, 60, 100\}$) and parameters with respect to the $\hat{\mathcal{D}}_{Ns}$ datasets. As expected, the solutions that yielded higher accuracies ($\mathcal{D}_{Ns}$) required correspondingly larger computational efforts, thus setting some design constraints to the deployment on embedded devices.

C. Embedded System Deployment.

The implementation on the Arduino Nano 33 BLE platform involved the 1-D CNN model that achieved the best generalization performance, i.e. the one trained with $\mathcal{D}_{100}$. The model held 2611 parameters (Table I), and was exported in both the float32 and the int8 quantized representations.

In the case of the int8 model, the average inference time on 120 data, having dimensions $4 \times 4 \times 100$, was about $26ms$, a considerable improvement over the $128ms$ required by the float32 model. The standard deviations were negligible in both cases. On average, the loss in generalization performance brought about by the int8 model with respect to the float32 model just affected $2/120$ of tested data. Both timing performances met the real-time constraint of 400ms [18] but, with a slight loss in generalization, the int8 quantized model lead to a saving of 100ms that could be used for data pre-processing.

VI. CONCLUDING REMARKS

The paper proposed the use of a 1-D CNN to support a 3-class touch modalities classification problem; the approach included two alternative pre-processing strategies, and covered eight datasets. A moving-average technique reduced the number of samples of the original dataset down to 100. The results proved that the proposed method outperformed state-of-the-art solutions in terms of both generalization performances and computational cost. Eventually, we deployed the best-performing architecture on a low-cost, resource-constrained device, by using both the 32-bit floating-point and the quantized, 8-bit integer representations. The resulting systems attained real-time performances ($< 400ms$) in the inference phase in both cases; with a marginal loss in generalization performance, the quantized int8 model also yielded a time saving of about $100ms$.

REFERENCES

[1] 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.

[2] 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.

[3] S.-y. Koo, J. G. Lim, and D.-s. Kwon, "Online touch behavior recognition of hard-cover robot using temporal decision tree classifier," in *RO-MAN 2008-The 17th IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2008, pp. 425–429.

[4] D. S. Tawil, D. Rye, and M. Velonaki, "Touch modality interpretation for an eit-based sensitive skin," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3770–3776.

[5] D. Silvera Tawil, D. Rye, and M. Velonaki, "Interpretation of the modality of touch on an artificial arm covered with an eit-based sensitive skin," *The International Journal of Robotics Research*, vol. 31, no. 13, pp. 1627–1641, 2012.

[6] J. Friedman, T. Hastie, R. Tibshirani *et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *Annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[7] M. Kaboli, A. Long, and G. Cheng, "Humanoids learn touch modalities identification via multi-modal robotic skin and robust tactile descriptors," *Advanced Robotics*, vol. 29, no. 21, pp. 1411–1425, 2015.
[8] M. M. Jung, M. Poel, R. Poppe, and D. K. Heylen, "Automatic recognition of touch gestures in the corpus of social touch," *Journal on multimodal user interfaces*, vol. 11, no. 1, pp. 81–96, 2017.
[9] D. Hughes, A. Krauthammer, and N. Correll, "Recognizing social touch gestures using recurrent and convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2315–2321.
[10] Y. Hu, S. M. Bejarano, and G. Hoffman, "Shadowsense: Detecting human touch in a social robot using shadow image classification," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 4, pp. 1–24, 2020.
[11] 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.
[12] H. Younes, A. Ibrahim, M. Rizk, and M. Valle, "Data oriented approximate k-nearest neighbor classifier for touch modality recognition," in *2019 15th Conference on Ph. D Research in Microelectronics and Electronics (PRIME)*. IEEE, 2019, pp. 241–244.
[13] H. Younes, M. Rizk, A. Ibrahim, and M. Valle, "Algorithmic-level approximate tensorial svm using high-level synthesis on fpga," *Electronics*, vol. 10, no. 2, p. 205, 2021.
[14] M. Alameh, A. Ibrahim, M. Valle, and G. Moser, "Dcnn for tactile sensory data classification based on transfer learning," in *2019 15th Conference on Ph. D Research in Microelectronics and Electronics (PRIME)*. IEEE, 2019, pp. 237–240.
[15] M. Alameh, Y. Abbass, A. Ibrahim, G. Moser, and M. Valle, "Touch modality classification using recurrent neural networks," *IEEE Sensors Journal*, 2021.
[16] Z. Yi, T. Xu, W. Shang, and X. Wu, "Touch modality identification with tensorial tactile signals: A kernel-based approach," *IEEE Transactions on Automation Science and Engineering*, 2021.
[17] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1d convolutional neural networks and applications: A survey," *Mechanical systems and signal processing*, vol. 151, p. 107398, 2021.
[18] A. Ibrahim and M. Valle, "Real-time embedded machine learning for tensorial tactile data processing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, no. 99, pp. 1–10, 2018.