6

Efficient Algorithms for Embedded Tactile Data Processing

Hamoud Younes[1,2,*], Mohamad Alameh[1], Ali Ibrahim[1,2], Mostafa Rizk[2] and Maurizio Valle[1]

[1]University of Genoa, Italy
[2]Lebanese International University, Lebanon
E-mail: hamoud.younes@edu.unige.it; mohammad.alameh@edu.unige.it; ali.ibrahim@edu.unige.it; mostafa.rizk@liu.edu.lb; maurizio.valle@unige.it
[*]Corresponding Author

This chapter presents a survey of the existing algorithms and tasks applied for tactile data processing. The presented algorithms and tasks include machine learning, deep learning, feature extraction, and dimensionality reduction. Moreover, this chapter provides guidelines for selecting appropriate hardware platforms for the algorithm's implementation. The algorithms are compared in terms of computational complexity and hardware implementation requirements. A touch modality classification problem is addressed as a case study: FPGA implementations of two algorithms k-Nearest Neighbor (KNN) and Support Vector Machine (SVM) are detailed and analyzed. Both algorithms provided real-time classification consuming 236 mW and 1.14 W, respectively. Such results can be improved with the use of approximate computing techniques that provide a trade-off between performance and hardware resources usage. Speedups up to 2× and 3.2× along with 30% and 41% power reduction are obtained for KNN and SVM implementations, respectively.

6.1 Introduction

Electronic skin (e-skin) is being incorporated in a wide range of systems such as Internet of Things (IoT), robotics, industrial automation, and

prosthetics [1]. E-skin is composed of an array of tactile sensors, an interface electronics, an embedded processing unit (EPU), and a communication interface [2]. The EPU is responsible for: (1) extracting and processing information from raw sensory data and (2) supporting intelligent tasks such as classification or regression based on sophisticated and complex algorithms (e.g. machine- and deep learning). To accomplish these tasks, the EPU must fulfill a set of requirements in terms of computational complexity and implementation requirements (size, latency, and power consumption) depending on the target application.

The data acquired from tactile sensors corresponds to an electrical stimulus. The latter varies according to the type of the sensing material, dimensionality, responsiveness, and structure of the sensor. Processing algorithms employed by the EPU should be able to decode and efficiently handle the acquired data. Although traditional processing algorithms (e.g. Fourier transforms) and machine/deep learning algorithms are effective for tactile data processing, their use is bounded by computational complexity and hardware implementation performance [3–5]. For instance, a neural network is considered as an efficient solution for classification problems but implementing it on hardware platforms imposes several challenges such as low time latency, low power, etc. [6–8]. Three surveys addressed such challenges. Sze et al presented the challenges faced in the embedded systems in Ref. [9], and how circuit designers are to address these challenges in Ref. [10]. In Ref. [11], the authors presented an overview of the existing techniques that enable efficient implementations of machine/deep learning algorithms. This chapter shows the steps to be followed to choose a convenient hardware platform for the target application.

The main of contributions of this chapter can be summarized as follows: (1) it presents preprocessing algorithms to extract data from tactile sensors, (2) it offers a survey of classification and regression algorithms that can be embedded into e-skins, (3) it provides an algorithmic level computational complexity study and guidelines for targeting convenient hardware platform for the hardware implementation, and (4) it tackles a case study on touch modality classification.

The rest of the chapter is organized as follows: Section 2 provides an overview of the algorithms used for tactile data processing. Moreover, an assessment of the presented algorithms is provided in terms of classification accuracy and computational complexity. Section 3 presents the different hardware platforms that have been used for the implementation of these

algorithms and the challenges faced. In Section 4, the FPGA implementation of two machine learning (ML) classifiers is presented as a case study for the embedded implementation of a touch modality classification. Finally, conclusions and future perspectives are highlighted in Section 5.

6.2 Tactile Data Processing Algorithms

Data processing algorithms presented in the literature could be divided into two categories: preprocessing and classification/regression. Preprocessing algorithms involve feature extraction and dimensionality reduction, while classification and regression algorithms are grouped into machine- and deep learning algorithms.

6.2.1 Data Preprocessing

Tactile data may be preprocessed to reduce noise and extract meaningful features. The extracted features could be (1) the variables that best describe the raw data and (2) the weights should be given for each variable. For instance, subsampling can be applied to a recorded touch reading to remove silent/noisy samples. Also, data obtained from certain taxels in the sensor patch can be considered in a pattern recognition problem. These taxels are the ones that provide reliable data (nonzero or unknown readings).

This section reports the algorithms presented in the literature for dimensionality reduction and feature extraction such as principal component analysis (PCA), independent component analysis (ICA), and linear discriminant analysis (LDA).

PCA is the base for multivariate data analysis (i.e. studying the effect of multiple variables on the output state) [12]. PCA is used for approximating data or reducing the dimensionality of the data e.g. representing data from X_n space in X_{n-k} space, where n and k are two positive integers. As a concrete example, if we have data with n features, then PCA helps to represent these data with $n - k$ features with the least possible losses. Figure 6.1 shows how PCA can be applied to reduce dimensionality from three dimensions (3D) to 2D (the figure has been generated using the data and code provided in Ref. [13]).

In Ref. [14], a finger-like shape tactile sensor has been used to collect data about fabric surfaces. Initially, fast Fourier transformation (FFT) was used to construct the original dataset, and then PCA was applied to compress the

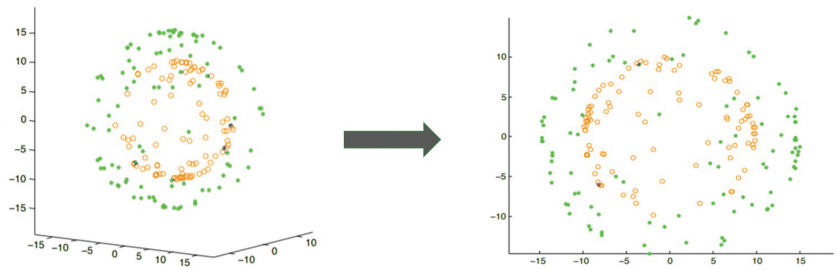

Figure 6.1 PCA example: 3D to 2D.

attribute data and extract feature information. In Ref. [15], kernel PCA [16] was used for low-resolution tactile image recognition for automated robotic assembly. Kernel PCA is a method to perform a nonlinear form of the PCA. It computes higher-order statistics among random variables while reducing the data dimensionality, thus being able to achieve the goal of both feature extraction and dimensionality reduction. Authors in Ref. [17] used local PCA [18] combined with a neural network to classify 16 household and toy objects. Local PCA is a nonlinear extension of the normal PCA. It has been used to obtain a less complex feature vector for the data obtained from tactile sensors mounted into a robotic arm.

ICA [19] can be seen as an extension of the PCA. It is a linear dimensionality reduction technique, which searches for the linear transformation that reduces or eliminates the linear dependency between elements of a random vector. An example of usage of ICA is the Cocktail Party Problem [20]. Spatial ICA has been adopted as a separation method that allows a robot to understand and interact with tactile information from multiple sources [21]. Figure 6.2 shows the procedure of tactile data separation from two objects using ICA along with time series clustering.

LDA shown in Figure 6.3 is yet another method for dimensionality reduction. It consists of finding the projection hyperplane that minimizes the variance within the same class, and maximizes the distance within the projected means of the classes [22].

Tactile images of deformable and nondeformable surfaces have been used for a classification problem in Ref. [23]. LDA has been used as a separation algorithm between six different surfaces with an accuracy rate of up to 95.5%. In Ref. [24], the authors have demonstrated the feasibility of using LDA for surface texture discrimination. Another use of LDA appears in Ref. [25] for terrain discrimination problems.

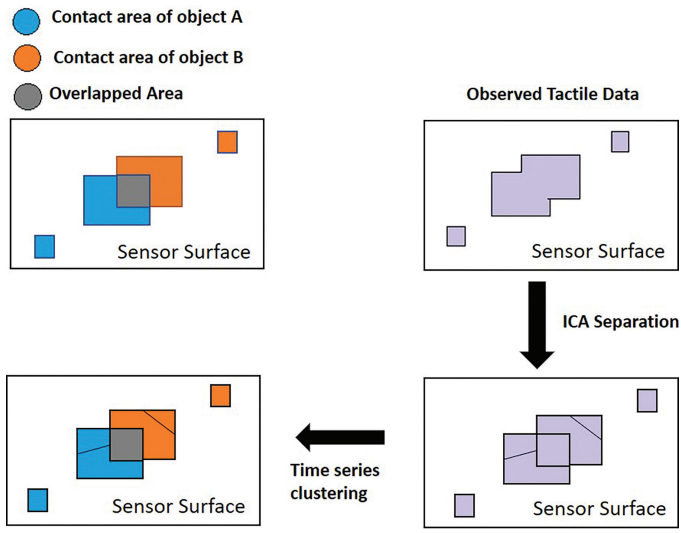

Figure 6.2 Procedure of tactile data separation using ICA.

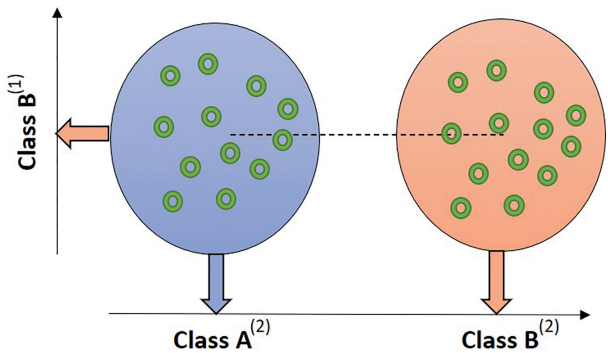

Figure 6.3 Linear discrimination analysis: (1) Bad projection and (2) Good projection.

6.2.2 Classification and Regression

6.2.2.1 Machine learning

Machine learning algorithms are an efficient solution for processing tactile data in various applications [26]. ML algorithms in general, can extract a complex, non linear input–output relationship based on learning by example approach. An ML algorithm is trained using a set of examples, where each example is described by a group of informative features. ML algorithms can support intelligent and predictive systems that can make accurate decisions on

unseen data. ML algorithms are categorized into supervised and unsupervised algorithms. Supervised algorithms are aware of labeled input and output data while unsupervised algorithms are fed with unlabeled input data. Some of the ML algorithms used for tactile data processing belongs to the supervised learning category [25–28]. These algorithms include Naïve Bayes (NB), Decision Tree (DT), K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Logitboost, etc.

NB is a probabilistic ML classifier that is based on Bayes' theorem given in Equation 6.1, with naïve independence assumptions between the features. It gives a quantitative approach to justify the evidence supporting a hypothesis i.e. the probability of occurrence of a certain action A given an action B has occurred. Generally, NB uses the Gaussian distribution parameters (e.g. mean, variance, etc.) on the dataset attributes for classification [29].

$$P(h/D) = \frac{P(D/h) \times P\,(h)}{P(D)} \tag{6.1}$$

where $P(h)$ is the probability that a hypothesis h holds; $P(D|h)$ is the probability of observing data D given a hypothesis h; $P(D)$ is the probability that data D will be observed; and $P(h|D)$ is the probability that a hypothesis h holds given the observed data D.

Authors in Ref. [30] have used an NB classifier in an industrial application for a vegetable grading robot. A PIC32 microcontroller was used to obtain pressure data from two piezoresistive flexible tactile sensors mounted on a two-fingered robotic arm. The arm was able to classify green, moderate, and ripe vegetables with an accuracy of 85%. In Ref. [31], a Denso robotic arm equipped with embedded strain gauge and polyvinylidene fluorides (PVDF) in two layers on the finger is used to classify five different materials based on their surface texture. Using an NB classifier, the system achieved an accuracy of $73 \pm 10\%$.

Another supervised ML algorithm that is widely used is the DT. DTs reflect human-level thinking by exploring the simple logic behind data interpretations. In a DT approach, a feature is represented by a node, a decision is represented by a branch, and an outcome is represented by a leaf. Two main algorithms are suggested to be used for building the DT algorithm: (1) Classification and Regression Trees (CART) that uses Gini Index G (a quantity that measures the degree of inequality in the distribution of a given data) as a metric and (2) Iterative Dichotomiser 3 (ID3) that uses entropy function $H(S)$ and information gain $IG(A, S)$ as metrics [30]. The three

metrics are given by the equations:

$$G = 1 - \sum_{t=0}^{t=k} P_t^2 \tag{6.2}$$

where k is the possible class value and P_t is the probability of occurrence of class t.

$$H(S) = \sum_{c \in C} -p(c)\log_2 p(c) \tag{6.3}$$

where S is the current dataset, C is the set of classes in S, and $p(c)$ is the proportion of the number of elements in C to the number of elements in S.

$$IG(A, S) = H(S) - \sum_{t \in T} p(t)H(t) \tag{6.4}$$

where T represents the subsets created from splitting S by attribute A such that $S = \cup_{t \in T} t$.

An autonomous humanoid robot from Aldebaran Robotics called NAO has been equipped with a 1.6 GHz Intel Atom CPU and suited with an artificial skin that has a multimodal tactile sensor [27]. The NAO has been trained to recognize nine different touch modalities (e.g. scratch, tickle, rub, etc.). Recognition of up to 96.8% was obtained using the DT algorithm. The NAO has been also trained using KNN and SVM classifiers reaching a classification accuracy of up to 95.1% and 96.75%, respectively.

A frequently used ML algorithm for classification problems is the KNN. KNN assigns a class C for an unseen query point q based on the class of the K-nearest points to q from a training set S. The distance from q to $p_i \in S$ can be Chebyshev, Manhattan, or Euclidean distance. The latter is the most popular metric as given by Equation 6.5:

$$d_{q,p_i} = \sqrt{\sum_{i=0}^{i=N} (q_{F_i} - p_{F_i})^2} \tag{6.5}$$

where F_i represents a feature of points q and p, and N is the size of the training set S.

Authors in Ref. [32] equipped an artificial fingertip with two perpendicular PVDF film sensors to acquire the surface roughness of eight standard solid nickel surfaces. Using a KNN classifier with $K = 9$, an average classification accuracy of $82.6 \pm 10\%$ has been attained. In Ref. [33], a tactile array sensor

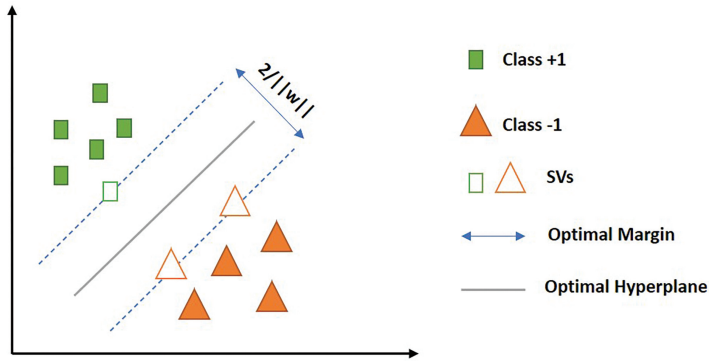

Figure 6.4 SVM hyperplane.

skin based on capacitive sensing technology is patched on the forearm of a humanoid called "Cody." Cody is used to obtain the mobility and compliance of 18 objects of different sizes. A KNN classifier with a variable value of K has been used. The classifier has differentiated between four different classes with a rate of 80% and $K = 2$. A higher classification accuracy of up to 91.43% was obtained for a binary classification problem with $K = 4$. Another binary classification problem for touch modality classification has been studied in Ref. [34]. A KNN classifier was trained to recognize two touch modalities: "rolling" and "sliding" recorded using a 4×4 tactile sensory array. The K parameter has varied to 3,5 and 7, and 10-fold cross-validation was applied to obtain credible results. The best classification accuracy was 89.6% for $K = 3$.

Another robust discriminative algorithm is SVM. SVM classification embraces the concept of a decision boundary that separates two different classes. This boundary is in the form of a hyperplane that is constructed in the training phase. The data points that lie on the boundary lines are called support vectors (SVs) as shown in Figure 6.4. These SVs are required for the classification phase.

Consider a training data labeled as $(x_i, y_i), i = 1, 2, \ldots, N$, $y_i \in \{-1, +1\}$ and $x_i \in R^d$. The hyperplane is the plane that separates the two classes of squares and triangles. Any pattern x that belongs to the hyperplane in the feature space can be described by Equation 6.6, where w is a normal vector to the hyperplane and b is a constant

$$wx + b = 0 \tag{6.6}$$

The data points are separated by the two hyperplanes described in Equations 6.7 and 6.8, thus, the main objective is to maximize the distance between them:

$$wx + b = +1 \tag{6.7}$$

$$wx + b = -1 \tag{6.8}$$

The optimum separation hyperplane conditions can be formulated into the expression given by Equation 6.9 where the distance $\parallel w \parallel$ needs to be minimized:

$$y_i(wx_i + b) \geq 1, i = 1, \ldots, N \tag{6.9}$$

In most ML classification problems, the training data can't be linearly separated in the original space. Thus, the input space is mapped to a higher dimensional one where linear separation is feasible. Such mapping is a computationally expensive task, especially for large-scale applications. Therefore, SVMs utilize kernel functions $K(x_i, x_j)$ that replace the inner product of the optimization problem in Equation 6.9 as given in Equation 6.10:

$$y_i(K(x_i, x_j) + b) \geq 1, i = 1, \ldots, N \tag{6.10}$$

The most common kernel functions used are linear, polynomial, sigmoid, and Gaussian radial basis function (RBF) [35]. In the SVM classification phase, a new unseen sample is classified according to the function given in Equation 6.11:

$$F(x) = \text{sign}\left(\sum \alpha_i y_i K(x_i, x) + b\right) \tag{6.11}$$

Five objects that have the same size, but different weights are covered with the most common surface textures (e.g. rough, sand, glass, etc.). These objects have been used to test the ability of an NAO robot to classify objects using an SVM classifier [28]. The paper claims a classification accuracy of up to 100%. In Ref. [36], a robotic hand with 5 fingers and 20 active degrees of freedom was equipped with a BioTac sensor to classify 20 daily used objects (e.g. ball, bottle, sponge, etc.). A least square SVM (LS-SVM) [37] classifier was adopted and a discrimination accuracy rate up to 97% was achieved. SVM has been also used in Ref. [32] for differentiating between eight nickel surfaces based on their roughness and it provided an accuracy rate up to 78.8 $\pm$ 14% using an RBF kernel.

Another set of ML algorithms that has been used for tactile data processing and reported in the literature includes locally weighted projection

Table 6.1 Computational complexity of machine learning algorithms

Algorithms	Applications	Training	Classification
Naïve Bayes	Classification	$O(nf)$	$O(f)$
Decision Tree	Classification/Regression	$O(n^2f)$	$O(f)$
SVM (kernel based)	Classification/Regression	$O(n^2f + n^3)$	$O(n_{SV}f)$
KNN	Classification/Regression	–	$O(nf)$
Linear Regression	Regression	$O(f^2n + f^3)$	$O(f)$
Random Forest	Classification/Regression	$O(n^2fn_{trees})$	$O(fn_{trees})$

regression [38], extreme learning machine [39], regularized extreme learning machine [40], and K-means clustering [41]. All these algorithms achieved a classification accuracy rate of $85 \pm 10\%$.

These algorithms will be implemented on different hardware devices/ platforms to be used as an EPU for the designed e-skin. When embedding ML algorithms in the e-skin, the implementation must maintain the hardware complexity, latency, and energy consumption as low as possible for portable and battery-powered devices. This means that an algorithm might support a certain application with high accuracy but contradicts with the available hardware space. Thus, a trade-off between the required accuracy and the available hardware space is to be considered. The trade-off is directly related to the computational complexity of these ML algorithms. Table 6.1 shows the computational complexity of the most commonly used algorithms in the Big-O notation [42], where n is the size of the training set, f is the number of features, n_{trees} is the number of trees, and n_{SV} is the number of support vectors.

The complexity given in Table 6.1 has been analyzed based on the degree of complexity provided by Figure 6.5. It is noticed that algorithms such as DT and SVM involve complex training phase that increases quadratically for a large number of training points. For linear regression (LR), the training phase complexity also increases quadratically with the number of features, which is usually less than the number of training points. Meanwhile, the DT, SVM, and LR classification phases are relatively less complex. For NB, the training phase is less complex compared to the SVM and DT with a low complexity classification phase too. This is due to the linear complexity compared to the quadratic one in the case of the SVM and DT. Similarly, the complexity of the classification phase of the KNN increases linearly with the increase in the number of training points and the number of features, and the same is

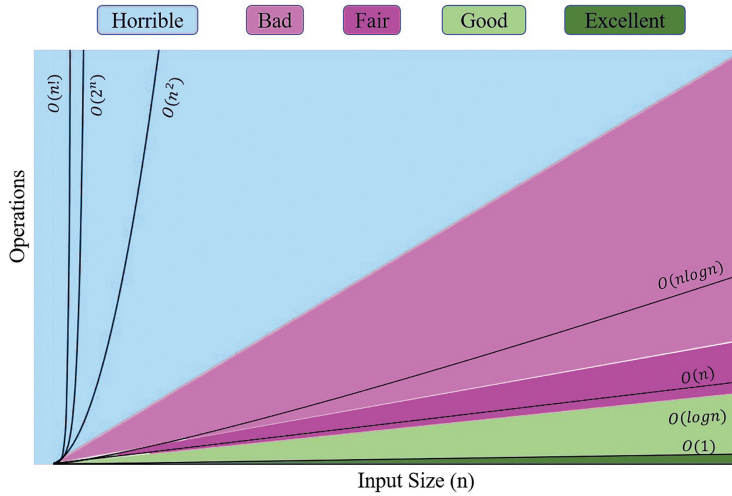

Figure 6.5 Big-O complexity.

observed from Equation 6.5. Although KNN doesn't have a separate training phase, it imposes a higher complexity compared to algorithms such as SVM ($n \gg nSV$).

6.2.2.2 Deep learning

Deep learning (DL) is a kind of artificial neural network (NN) where the network has more hidden layers inside it [43]. Usually, DL is used as a classification and feature extraction method at the same time (especially in image processing); so, no handcrafting of features is required [44, 45].

A convolutional neural network (CNN) is a deep neural network – the name is derived from the convolutional layers used in this network – that is usually used in image processing. In tactile sensing, different works used DNN or CNN for tactile data classification.

In Ref. [7], the CNN is used to evaluate four attributes for 23 gel-like foods: elasticity, smoothness, stickiness, and granularity. Four separate CNNs were trained where the input is a time series image coming from a pressure sensor. The image is 44×44 pixels (without the boundaries) in size. The network has four outputs, and each one corresponds to the sensory evaluation of a single attribute e.g. elasticity.

A high-resolution tactile sensor attached to a robotic arm was used to collect pressure maps of 22 daily-life objects, of 28×50 pixels each in Ref. [46]. Different CNNs were used, some pretrained on millions of images

[47], and others were built from the scratch. A combination of two CNNs was used for tactile data classification: one CNN for sequential data coming from tactile sensors $(32 \times 32 \times 32)$ and another coming from visual interesting points (25×3) for identifying eight simulated objects: plane, bird, car, chair, hand, vase, quadruped, and head. Combining these two networks showed an improvement compared to the results obtained from tactile data only without visual guidance [48].

In Ref. [49], benchmark image processing CNNs were used to classify tensorial data collected from 4×4 tactile sensors by transfer learning. The CNNs were trained on a large number of images, and then retrained on synthetic images coming from tactile sensors to classify three touch modalities: brushing, rolling, and sliding.

6.3 Embedded Processing System

6.3.1 Hardware Platforms

The hardware platform must be able to handle the complexity of the algorithm while achieving the expected performance in terms of time latency and energy consumption. A wide selection of hardware devices and platforms maybe used to implement the tactile processing algorithms. Some sound and widely used devices include field programmable gate array (FPGA), graphics processing unit (GPU), microcontroller unit (MCU), parallel ultralow-power platform (PULP), tensor processing unit (TPU), application-specific integrated circuit (ASIC), and platforms include Raspberry Pie, ZedBoard, Zynqberry, Python Productivity for Zynq (PYNQ), etc.

The available hardware devices and platforms differ in size, target programming language, area utilization (LUT, FF, DSP, BRAM, etc.), maximum operating frequency, etc. Table 6.2 presents the common characteristics of the most used hardware devices and platforms related to the variety of machine- and deep learning applications.

Taking into consideration the information presented in Table 6.2, an FPGA is suitable for implementing simple/moderate ML algorithms such as the LR or DT, while a neural network is the best fit on a GPU. Similarly, a more complex algorithm such as the KNN/SVM can be implemented on the FPGA, PULP, or ZedBoard/Zynqberry but the best choice depends on the application requirements. For example, choosing the GPU for a wearable device (e.g. smart watch) is not feasible, and the same can be said for implementing AI training on a microcontroller. Besides, the development

Table 6.2 Comparison of hardware devices/platforms

Types	Names	Framework/ Programming Languages	Strengths	Weaknesses
Device	FPGA	VHDL, Verilog, C/C++ with OpenCL, SDAccel, HLS	High performance per watt, parallelism	Not suited for floating-point operations, long development time, programming difficulty
	GPU	OpenCL, NVIDIA CUDA, C/C++, Java, Python	Massive processing power for image, video, and signal processing	High power consumption, need for API frameworks to take advantage of parallelism
	ASIC	Application-specific, ex: TensorFlow for TPUs, tools from manufactures	Optimum combination of performance and power consumption	High cost, long development time, not configurable
Platform	PULP	C language only	Low power consumption, tunable performance, Open source	Low size on-chip memory, long development time
	ZedBoard/Zynqberry	– adopts characteristics of FPGA and ARM processors – Pluses: Ability to use FPGA as a hardware accelerator, Linux Development		
	PYNQ	– adopts characteristics of FPGA and ARM processors – Pluses: Python Programming, Arduino and Raspberry Pie shield connectors		

time is a key issue to consider, especially for complex ML/DL algorithms such as implementing a CNN using VHDL language. All these considerations are to be discussed before selecting an appropriate hardware platform.

In general, the GPU is one of the best performers for fast ML processing. This is due to the available high memory bandwidth and a large number of processing cores. On the other hand, GPU computations can be about four times more expensive than CPU computations. So, if the gain in speedup is

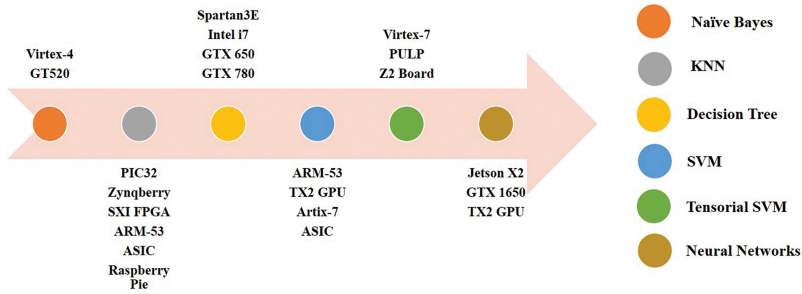

Figure 6.6 Hardware platforms used for different ML and deep learning algorithms.

not relative to that cost, CPUs can be more suitable than GPUs. Moreover, in each ML application, the training and testing implementations are not correlated. Hence, the GPU can be used for neural network training, while preserving the CPU for testing purposes. The same analogy can be viewed for all the hardware platforms. Figure 6.6 shows the hardware platforms reported in the literature that are used with the increased complexity of ML algorithms.

Although Section 2 provided an overview of machine- and deep learning algorithms that have been used for tactile data processing, the implementation of these algorithms on hardware platforms is still a challenge. The hardware implementations of the above-highlighted algorithms will be assessed even for different applications i.e. not limited to tactile data. This is because of the lack of such implementations for tactile data processing. Moreover, this assessment will help in studying the feasibility of implementing the algorithm itself on each hardware platform.

Table 6.3 presents some the hardware implementations of different ML algorithms. The implementation of Naïve Bayes has been carried out on Virtex-4 FPGA consuming a total of 2% occupied area for handwriting recognition problem of 70,000 samples [50]. In Ref. [51], a parallel architecture of NB has been implemented using the GPUs. For document classification problem, a speedup up to $34\times$ and $11\times$ can be achieved compared to sequential and parallel versions, respectively, using CPU. The GPU used was GeForce GT520 2GB graphics card and the number of used documents is 861,454. A pipelined DT implementation on FPGA has been presented in Ref. [52]. The development board used was Digilent Nexys-2 Spartan-3E FPGA board. The complete DT model required 6442 LUTs, 5336 FFs, and 22 block RAMs, which resemble 62% of area utilization. For a tree with

Table 6.3 Embedded machine learning on different hardware devices/platforms

Algorithms	Devices/Platforms	Implementation Results
Naïve Bayes	Virtex-4 FPGA	2% area utilization
	GeForce GT520 GPU	34× speedup compared to Intel Core 17-2600 CPU operating at 3,40 GHz,
Decision Tree	Digilent Nexys-2 Spartan-3E FPGA	62% area utilization Time latency of 220 ns for a tree with 100 tuples
	GeForce GTX 650 Ti	485× speedup compared to quadcore processor (Intel Core i7-870, 8M Cache, 2.93 GHz),
SVM	ARM 53 processor	1530 mW power consumption running at 990 MHz
	Jetson TX2 GPU	29× speedup compared to ARM 53 Processor 2090 mW power consumption running at 854 MHz
	Artix-7 FPGA	6× increase in throughput compared to Raspberry Pi 3B 1/5 power reduction compared to Raspberry Pi 3B
	Virtex-7 FPGA	30% area utilization A peak performance of 302 G-ops while consuming 1.14 W
	PULP	34x speedup compared to ARM Cortex M4 running at 168 MHz Power consumption $<$150 mW
	ASIC	0.3 mm^2 area utilization using 65 nm CMOS technology Energy of 13.4 nJ running at 5 Hz nominal frequency 42x energy efficient more than Artix-7 FPGA
KNN	ASIC	0.16 mm^2 area utilization using 65 nm CMOS technology Energy of 0.31 nJ running at 59 Hz nominal frequency 12x energy efficient more than Artix-7 FPGA
	Artix-7 FPGA	5x increase in throughput compared to Raspberry Pi 3B 1/4 power reduction compared to Raspberry Pi 3B
	ARM 53 processor	Power consumption of 1480 mW running at 990 MHz
	Jetson TX2 GPU	29x speedup compared to ARM 53 Processor Power consumption of 2120 mW running at 854 MHz

100 tuples, a time latency of 220 ns was recorded. A set of NVIDIA boards was used to evaluate the performance of DT models for large-scale data [53]. A maximum mean speedup of $585\times$ for a dataset of 10 M instances was obtained. The authors in Ref. [54] have implemented the KNN and SVM classifiers on ARM 53 processors, Jetson TX2 GPU, Artix-7 FPGA, and ASIC. The proposed SVM and KNN classifiers on the ASIC platform occupy an area of 0.17 mm^2 and 0.3 mm^2 while dissipating 39.4 mW and 76.9 mW, respectively. The experimental results showed that the use of FPGA and ASIC lead to the highest throughput (decision/s) as well as the lowest power consumption. The obtained results were also superior to the one obtained when using Raspberry Pi.

For a touch modality classification problem, two architectures for tensorial SVM [2] and DCNN [49] have been applied on Virtex-7 FPGA and Jetson TX2 development board from NVIDIA, respectively. The FPGA implementation provided a real-time classification and a power consumption of 1.14 W. In Ref. [55], a tactile data decoding module using SVM based tensor kernel algorithm for touch modalities was implemented on PULP [56]. The decoding module ensured a power consumption of less than 150 mW for a wearable device requirement. The proposed implementation runs $34\times$ faster than an ARM Cortex M4 running at 168 MHz at the same power consumption. In Ref. [57], an Enclustra SX1 FPGA was used to process the tactile data obtained by a novel Hex-O-Skin, while a PIC32 MCU has been utilized to obtain the pressure data from two PVDF tactile sensors [30]. Another 32-bit microcontroller, Teensy 3.2, was adopted as the main unit of a Vibrotactile Stimulation system [58].

6.4 Case Study: Touch Modality Classification

The touch modality classification problem has been the focus of several works in the literature e.g. [26, 27, 39]. We have surveyed the literature for the different ML algorithms used for touch modality classification. Based on this survey, most of the works reported that the SVM and KNN are the most effective ML algorithms to deal with this problem.

This section introduces the experimental setup used for touch modality classification in terms of the used dataset, preprocessing techniques and the performance of the algorithms in terms of classification accuracy. Then, the FPGA implementation of these two algorithms is presented and analyzed.

6.4.1 Experimental Setup

The touch modality problem we are targeting is the binary classification problem i.e. "sliding a finger" vs "rolling a washer" [26]. The dataset used contains data for 70 participants. Each participant performed a touch on a 4×4 tactile sensor for 10 seconds on both the horizontal and vertical directions. Thus, the final dataset contained 280 patterns. Using a 3 kHz sampling frequency, each touch was presented as a tensor of size $4 \times 4 \times 30,000$, where 30,000 raw samples were recorded from the 4×4 sensor during the 10 s duration. KNN [34] and SVM [26] classifiers based on the tensorial representation of input were recently proposed for the binary classification problem.

- KNN classifier: The authors have applied a feature extraction process on the initial dataset. First, the samples outside the range of 3.5 to 7 s were removed as they involved static movement or noisy information. Then, the mean of each 30,000 samples was calculated, resulting in a tensor of size $4 \times 4 \times 1$. Several simulation scenarios were studied and reported in Table 6.4 [34].
- SVM classifier: The authors have considered the data acquired in the first 7 out of 10 s using the same sampling frequency of 3 kHz. This resulted in a tensor size of $4 \times 4 \times 21000$. Such tensor size imposed an impractical computational task. Thus, the amount of energy provided by each single element of the sensor was analyzed. This task showed that only a portion of the 21,000 elements carry actual information. Then, a subsampling strategy was applied to find the best tensor 3rd dimension size D. The different simulations carried out are summarized in Table 6.5 [26] where λ, σ, and α represent the kernel parameters used.

The results in Tables 6.4 and 6.5 justify the use of KNN and SVM to support the touch modality classification problem with their classification accuracy. Recalling that these ML algorithms will be embedded into an electronic skin, the following section details the implementation of these

Table 6.4 KNN classification results

	Classification Accuracy (%)		
Training Dataset Size	K = 3	K = 5	K = 7
80% split	84	80.3	79
85% split	86	83.3	81
90% split	82.1	82.1	82.1
10-fold Cross Validation	89.6	89.3	89

Table 6.5 SVM classification results

Simulation Scenario	Classification Accuracy (%)		
	D = 20	D = 50	D = 100
Run #1	85 ($\lambda = 0.1$, $\sigma = 1$, $\alpha = Qz/2$)	83.5 ($\lambda = 0.1$, $\sigma = 1$, $\alpha = Qz/2$)	83.5 ($\lambda = 0.1$, $\sigma = 1$, $\alpha = Qz/2$)
Run #2	87.5 ($\lambda = 1$, $\sigma = 2^4$, $\alpha = Qz/2$)	85 ($\lambda = 10$, $\sigma = 2^3$, $\alpha = Qz/2$)	90 ($\lambda = 1$, $\sigma = 2^{-1}$, $\alpha = 0$)
Run #3	80 ($\lambda = 0.1$, $\sigma = 2^1$, $\alpha = Qz/2$)	87.5 ($\lambda = 1$, $\sigma = 2^2$, $\alpha = Qz/2$)	90 (λ=1, σ=2^2, α= Qz/2)

algorithms on FPGA by exploring the hardware area occupied, time latency, and power consumption.

6.4.2 Implementation Details

The FPGA implementation of the tensorial SVM classifier is reported in Ref. [2]. The paper proposed two different architectures: cascaded and parallel to reach an adequate trade-off between real-time functionality and hardware resources. The parallel implementation reported 1.14 W power consumption, while achieving a peak performance of 302 G-ops. A tensor size of $8 \times 8 \times 20$ is used to represent the input data.

As for the KNN, the implementation is carried out using high-level synthesis (HLS) on Zynqberry [59]. The KNN classifies the unseen sample by executing the following steps: (1) distance calculation from the unseen sample to all the training samples, (2) the calculated distances are sorted in ascending order, and (3) the $K = 3$ neighbours with the smallest distances are chosen, and the output class is the class of the majority of the three neighbors. The KNN classifier was coded in C++ and optimized using Vivado HLS directives. Then, it was exported as an RTL intellectual property (IP) block. The IP was imported into Vivado to obtain the implementation report. The report showed that the KNN classifier consumes 236 mW while classifying a new sample within 1 ms.

Tables 6.6 and 6.7 summarize the outcome of the implementations on FPGA. KNN was implemented on the Zynqberry Platform (XC7Z010ICLG225-1L FPGA) operating at 100 MHz. SVM implementation was carried out on Virtex-7 XC7VX980T operating at 120 MHz.

The obtained results illustrate the feasibility of the implementation of the most used ML algorithms for tactile data processing. For embedding these algorithms into e-skin, some observations must be considered. For KNN,

Table 6.6 Implementation details for SVM on FPGA

Algorithm	Tensor Size	Training Tensors	Occupied Area	Time Latency	Power Consumption
SVM	$8 \times 8 \times 20$	100	13%	<200 ms	1.14 W

Table 6.7 Implementation details for KNN on FPGA

Algorithm	Tensor Size	Training Tensors	Occupied Area	Time Latency	Power Consumption
KNN	$4 \times 4 \times 1$	280	3%	<3 ms	236 mW

although the obtained power consumption is low, this could increase dramatically if the training size is large. As for SVM, the power consumption is relatively high, and the occupied area may not be suitable for e-skin size. It is worth noting that these two algorithms use a tensor-based input data, which is reported to preserve the initial information of the touch [60].

Taking these results into consideration with the complexity study in Table 6.1, and the implementations reported in the literature, several solutions were proposed to decrease the complexity of the embedded ML implementations [54, 55]. One of the effective solutions is the use of approximate computing techniques (ACTs). The authors in Ref. [61] have presented an approach for applying algorithmic level ACTs for the discussed KNN and SVM implementations. A circuit-level ACT is presented in Ref. [62] that utilizes the use of inexact accumulators for ML algorithms. For instance, the classification phase of the KNN and SVM can be accelerated up to $2\times$ and $3.2\times$ while achieving 30% and 41% power reductions, respectively, when applying algorithmic-level ACTs. Also, power savings up to 69% is achieved when the inexact accumulators are used in the SVM classifier implementation.

6.5 Conclusion

This chapter introduced the state-of-the-art processing algorithms used for tactile data processing. Starting with feature extraction algorithms to obtain a meaningful representation of data, the chapter proceeds to machine- and deep learning algorithms used for classification and regression applications. Then, several existing hardware platforms were studied as candidates for embedding the algorithms in the e-skin. Finally, two of the presented algorithms (KNN and SVM) were adopted for a touch modality classification problem. Similarly, the implementation of these algorithms was conducted on

two different platforms: Virtex-7 and Zynqberry. The obtained results showed a low latency and area size while delivering considerably challenging power consumption in the case of the SVM. Such a challenge has been tackled using approximate computing, reaching a speedup of up to $3.2\times$ with 41% power reductions, without affecting the quality of the target application.

References

[1] L. Seminara et al., 'Towards integrating intelligence in electronic skin', *Mechatronics*, vol. 34, pp. 84–94, Mar. 2016, doi: 10.1016/j.mechatronics.2015.04.001.

[2] A. Ibrahim and M. Valle, 'Real-time embedded machine learning for tensorial tactile data processing', *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 65, no. 11, pp. 3897–3906, Nov. 2018, doi: 10.1109/TCSI.2018.2852260.

[3] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, 'Erratum to: A survey of machine learning for big data processing', *EURASIP J. Adv. Signal Process.*, vol. 2016, no. 1, p. 85, Dec. 2016, doi: 10.1186/s13634-016-0382-7.

[4] U. S. Shanthamallu, A. Spanias, C. Tepedelenlioglu, and M. Stanley, 'A brief survey of machine learning methods and their sensor and IoT applications', in *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, Larnaca, Aug. 2017, pp. 1–8, doi: 10.1109/IISA.2017.8316459.

[5] S. Pouyanfar et al., 'A survey on deep learning: Algorithms, techniques, and applications', *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–36, Jan. 2019, doi: 10.1145/3234150.

[6] Z. Hajduk, 'Reconfigurable FPGA implementation of neural networks', *Neurocomputing*, vol. 308, pp. 227–234, Sep. 2018, doi: 10.1016/j.neucom.2018.04.077.

[7] A. Shibata, A. Ikegami, M. Nakauma, and M. Higashimori, 'Convolutional neural network-based estimation of gel-like food texture by a robotic sensing system', *Robotics*, vol. 6, no. 4, p. 37, 2017.

[8] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, 'Optimizing FPGA-based accelerator design for deep convolutional neural networks', in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays – FPGA'15*, Monterey, California, USA, 2015, pp. 161–170, doi: 10.1145/2684746.2689060.

[9] V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, and Z. Zhang, 'Hardware for machine learning: Challenges and opportunities', in *2017 IEEE Custom Integrated Circuits Conference (CICC)*, Austin, TX, Apr. 2017, pp. 1–8, doi: 10.1109/CICC.2017.7993626.

[10] V. Sze, 'Designing hardware for machine learning: The important role played by circuit designers', *IEEE Solid-State Circuits Mag.*, vol. 9, no. 4, pp. 46–54, 2017, doi: 10.1109/MSSC.2017.2745798.

[11] M. Osta, M. Alameh, H. Younes, A. Ibrahim, and M. Valle, 'Energy efficient implementation of machine learning algorithms on hardware platforms', presented at the *26th IEEE International Conference on Electronics Circuits and Systems*, pp. 21–24, Genova, Italy, Nov. 2019.

[12] S. Wold, K. Esbensen, and P. Geladi, 'Principal component analysis', *Chemom. Intell. Lab. Syst.*, vol. 2, no. 1–3, pp. 37–52, 1987.

[13] L. Derksen, 'Visualising high-dimensional datasets using PCA and t-SNE in Python', *Medium*, Apr. 29, 2019. https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b (accessed June 02, 2020).

[14] H. Hu, Y. Han, A. Song, S. Chen, C. Wang, and Z. Wang, 'A finger-shaped tactile sensor for fabric surfaces evaluation by 2-Dimensional active sliding touch', *Sensors*, vol. 14, no. 3, pp. 4899–4913, Mar. 2014, doi: 10.3390/s140304899.

[15] Y.-H. Liu, Y.-T. Hsiao, W.-T. Cheng, Y.-C. Liu, and J.-Y. Su, 'Low-resolution tactile image recognition for automated robotic assembly using kernel PCA-based feature fusion and multiple kernel learning-based support vector machine', *Math. Probl. Eng.*, vol. 2014, pp. 1–11, 2014, doi: 10.1155/2014/497275.

[16] B. Schölkopf, A. Smola, and K.-R. Müller, 'Kernel principal component analysis', in *International Conference on Artificial Neural Networks*, Lausanne, Switzerland, Oct. 1997, pp. 583–588.

[17] M. Schopfer, H. Ritter, and G. Heidemann, 'Acquisition and application of a tactile database', in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, Apr. 2007, pp. 1517–1522, doi: 10.1109/ROBOT.2007.363539.

[18] A. Weingessel and K. Hornik, 'Local PCA algorithms', *IEEE Trans. Neural Netw.*, vol. 11, no. 6, pp. 1242–1250, Nov. 2000, doi: 10.1109/72.883408.

[19] P. Comon, 'Independent component analysis. A new concept?', *Signal Process.*, vol. 36, no. 3, pp. 287–314, Apr. 1994, doi: 10.1016/0165-1684(94)90029-9.

[20] S. Haykin and Z. Chen, 'The cocktail party problem', *Neural Comput.*, vol. 17, no. 9, pp. 1875–1902, 2005.

[21] K. Lee, T. Ikeda, T. Miyashita, H. Ishiguro, and N. Hagita, 'Separation of tactile information from multiple sources based on spatial ICA and time series clustering', in *2011 IEEE/SICE International Symposium on System Integration (SII)*, Kyoto, Japan, Dec. 2011, pp. 791–796, doi: 10.1109/SII.2011.6147549.

[22] P. Xanthopoulos, P. M. Pardalos, and T. B. Trafalis, 'Linear discriminant analysis', in *Robust Data Mining*, P. Xanthopoulos, P. M. Pardalos, and T. B. Trafalis, Eds. New York, NY: Springer, 2013, pp. 27–33.

[23] M. Pal, A. Khasnobish, A. Konar, D. N. Tibarewala, and R. Janarthanan, 'Classification of deformable and non-deformable surfaces by tactile image analysis', in *Proceedings of The 2014 International Conference on Control, Instrumentation, Energy and Communication (CIEC)*, Calcutta, India, Jan. 2014, pp. 626–630, doi: 10.1109/CIEC.2014.6959165.

[24] H. Nguyen et al., 'Dynamic texture decoding using a neuromorphic multilayer tactile sensor', in *2018 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Cleveland, OH, Oct. 2018, pp. 1–4, doi: 10.1109/BIOCAS.2018.8584826.

[25] W. Kalas, 'Tactile sensing for ground classification', *J. Autom. Mob. Robot. Intell. Syst.*, vol. 7, no. 2, pp. 18–23, 2013.

[26] P. Gastaldo, L. Pinna, L. Seminara, M. Valle, and R. Zunino, 'Computational intelligence techniques for tactile sensing systems', *Sensors*, vol. 14, no. 6, pp. 10952–10976, June 2014, doi: 10.3390/s140610952.

[27] M. Kaboli, A. Long, and G. Cheng, 'Humanoids learn touch modalities identification via multi-modal robotic skin and robust tactile descriptors', *Adv. Robot.*, vol. 29, no. 21, pp. 1411–1425, Nov. 2015, doi: 10.1080/01691864.2015.1095652.

[28] M. Kaboli, P. Mittendorfer, V. Hugel, and G. Cheng, 'Humanoids learn object properties from robust tactile feature descriptors via multi-modal artificial skin', in *2014 IEEE-RAS International Conference on Humanoid Robots*, Madrid, Spain, Nov. 2014, pp. 187–192, doi: 10.1109/HUMANOIDS.2014.7041358.

[29] D. Xu, G. E. Loeb, and J. A. Fishel, 'Tactile identification of objects using Bayesian exploration', in *2013 IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013, pp. 3056–3061, doi: 10.1109/ICRA.2013.6631001.

[30] I. Bandyopadhyaya, D. Babu, A. Kumar, and J. Roychowdhury, 'Tactile sensing based softness classification using machine learning', in *2014*

IEEE International Advance Computing Conference (IACC), Gurgaon, India, Feb. 2014, pp. 1231–1236, doi: 10.1109/IAdCC.2014.6779503.

[31] N. Jamali and C. Sammut, 'Majority voting: Material classification by tactile sensing using surface texture', *IEEE Trans. Robot.*, vol. 27, no. 3, pp. 508–521, June 2011, doi: 10.1109/TRO.2011.2127110.

[32] Z. Yi, Y. Zhang, and J. Peters, 'Bioinspired tactile sensor for surface roughness discrimination', *Sens. Actuators Phys.*, vol. 255, pp. 46–53, Mar. 2017, doi: 10.1016/j.sna.2016.12.021.

[33] T. Bhattacharjee, J. M. Rehg, and C. C. Kemp, 'Haptic classification and recognition of objects using a tactile sensing forearm', in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura-Algarve, Portugal, Oct. 2012, pp. 4090–4097, doi: 10.1109/IROS.2012.6386142.

[34] H. Younes, A. Ibrahim, M. Rizk, and M. Valle, 'Data oriented approximate K-nearest neighbor classifier for touch modality recognition', presented at the 15th Conference on PhD Research in Microelectronics and Electronics, Lausanne, Switzerland, 2019.

[35] S. M. Afifi, H. Gholam Hosseini, and R. Sinha, 'Hardware implementations of SVM on FPGA: A state-of-the-art review of current practice', *International Journal of Innovative Science, Engineering & Technology*, vol. 2, no. 11, pp. 2348–7968, Nov. 2015.

[36] M. Kaboli, R. Walker, and G. Cheng, 'Re-using prior tactile experience by robotic hands to discriminate in-hand objects via texture properties', in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, May 2016, pp. 2242–2247, doi: 10.1109/ICRA.2016.7487372.

[37] J. A. K. Suykens and J. Vandewalle, 'Least squares support vector machine classifiers', *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 1999, doi: 10.1023/A:1018628609742.

[38] Z. Su et al., 'Force estimation and slip detection/classification for grip control using a biomimetic tactile sensor', in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Seoul, South Korea, Nov. 2015, pp. 297–303, doi: 10.1109/HUMANOIDS.2015.7363558.

[39] P. Gastaldo, L. Pinna, L. Seminara, M. Valle, and R. Zunino, 'A tensor-based pattern-recognition framework for the interpretation of touch modality in artificial skin systems', *IEEE Sens. J.*, vol. 14, no. 7, pp. 2216–2225, July 2014, doi: 10.1109/JSEN.2014.2320820.

[40] S. Decherchi, P. Gastaldo, R. S. Dahiya, M. Valle, and R. Zunino, 'Tactile-data classification of contact materials using computational intelligence', *IEEE Trans. Robot.*, vol. 27, no. 3, pp. 635–639, June 2011, doi: 10.1109/TRO.2011.2130030.

[41] S. Luo, W. Mou, M. Li, K. Althoefer, and H. Liu, 'Rotation and translation invariant object recognition with a tactile sensor', in *IEEE SENSORS 2014 Proceedings*, Valencia, Nov. 2014, pp. 1030–1033, doi: 10.1109/ICSENS.2014.6985179.

[42] 'Big-O Algorithm Complexity Cheat Sheet (Know Thy Complexities!) @ericdrowell'. https://www.bigocheatsheet.com/ (accessed Jan. 05, 2020).

[43] J. Schmidhuber, 'Deep learning in neural networks: An overview', *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015, doi: 10.1016/j.neunet.2014.09.003.

[44] A. E. Hassanien, A. Darwish, and C. L. Chowdhary, Eds., *Handbook of Research on Deep Learning Innovations and Trends*. IGI Global, Hershey, Pennsylvania 2019.

[45] D. J. Hemanth and V. V. Estrela, Eds., *Deep Learning for Image Processing Applications*. Amsterdam: IOS Press, 2017.

[46] J. M. Gandarias, A. J. García-Cerezo, and J. M. Gómez-de-Gabriel, 'CNN-based methods for object recognition with high-resolution tactile sensors', *IEEE Sens. J.*, vol. 19, no. 16, pp. 6872–6882, Aug. 2019, doi: 10.1109/JSEN.2019.2912968.

[47] O. Russakovsky et al., 'ImageNet Large Scale Visual Recognition Challenge', *ArXiv14090575 Cs*, Jan. 2015, [Online]. Available at http://arxiv.org/abs/1409.0575 [accessed Apr. 06, 2020].

[48] G. Rouhafzay and A.-M. Cretu, 'An application of deep learning to tactile data for object recognition under visual guidance', *Sensors*, vol. 19, no. 7, p. 1534, Jan. 2019, doi: 10.3390/s19071534.

[49] 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)*, Lausanne, Switzerland, July 2019, pp. 237–240, doi: 10.1109/PRIME.2019.8787748.

[50] H. Meng, K. Appiah, A. Hunter, and P. Dickinson, 'FPGA implementation of Naive Bayes classifier for visual object recognition', in *CVPR 2011 WORKSHOPS*, Colorado Springs, CO, USA, June 2011, pp. 123–128, doi: 10.1109/CVPRW.2011.5981831.

[51] G. Andrade et al., 'GPU-NB: A Fast CUDA-based implementation of Naive Bayes', in *2013 25th International Symposium on Computer Architecture and High Performance Computing*, Porto de Galinhas, Pernambuco, Brazil, Oct. 2013, pp. 168–175, doi: 10.1109/SBAC-PAD.2013.16.

[52] F. Saqib, A. Dutta, J. Plusquellic, P. Ortiz, and M. S. Pattichis, 'Pipelined decision tree classification accelerator implementation in FPGA (DT-CAIF)', *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 280–285, Jan. 2015, doi: 10.1109/TC.2013.204.

[53] K. Jurczuk, M. Czajkowski, and M. Kretowski, 'Evolutionary induction of a decision tree for large-scale data: a GPU-based approach', *Soft Comput.*, vol. 21, no. 24, pp. 7363–7379, Dec. 2017, doi: 10.1007/s00500-016-2280-1.

[54] N. Attaran, A. Puranik, J. Brooks, and T. Mohsenin, 'Embedded low-power processor for personalized stress detection', *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 65, no. 12, pp. 2032–2036, Dec. 2018, doi: 10.1109/TCSII.2018.2799821.

[55] M. Magno, A. Ibrahim, A. Pullini, M. Valle, and L. Benini, 'An energy efficient E-skin embedded system for real-time tactile data decoding', *J. Low Power Electron.*, vol. 14, no. 1, pp. 101–109, Mar. 2018, doi: 10.1166/jolpe.2018.1537.

[56] 'PULP platform'. https://www.pulp-platform.org/ (accessed Jan. 05, 2020).

[57] P. Mittendorfer and G. Cheng, 'Humanoid multimodal tactile-sensing modules', *IEEE Trans. Robot.*, vol. 27, no. 3, pp. 401–410, June 2011, doi: 10.1109/TRO.2011.2106330.

[58] P. Asman, T. Jiang, M. Ozturk, J. Reyna, and N. F. Ince, 'A low-cost microcontroller based stimulation system to study sensory processing', in *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, San Francisco, CA, USA, Mar. 2019, pp. 883–886, doi: 10.1109/NER.2019.8716944.

[59] 'TE0726 Zynqberry Demo1 – Public Docs – Trenz Electronic Wiki'. https://wiki.trenz-electronic.de/display/PD/TE0726+Zynqberry+Demo1 (accessed Jan. 05, 2020).

[60] M. Signoretto, L. De Lathauwer, and J. A. K. Suykens, 'A kernel-based framework to tensorial data analysis', *Neural Netw.*, vol. 24, no. 8, pp. 861–874, Oct. 2011, doi: 10.1016/j.neunet.2011.05.011.

[61] H. Younes, A. Ibrahim, M. Rizk, and M. Valle, 'Algorithmic level approximate computing for machine learning classifiers', presented at

the *26th IEEE International Conference on Electronics Circuits and Systems*, Genova, Italy, Nov. 2019.

[62] Y. Zhou, J. Lin, and Z. Wang, 'Energy efficient SVM classifier using approximate computing', in *2017 IEEE 12th International Conference on ASIC (ASICON)*, Guiyang, Oct. 2017, pp. 1045–1048, doi: 10.1109/ASICON.2017.8252658.