Sensor Applications

# Trade-off between Accuracy and Computational Cost with Neural Architecture Search: a Novel Strategy for Tactile Sensing Design

Christian Gianoglio[1*], Edoardo Ragusa[1*], Paolo Gastaldo[1], and Maurizio Valle[1**]

[1] *Electrical, Electronics, and Telecommunication Eng. and Naval Architecture Department (DITEN), University of Genoa, Genoa, Italy*
[*] *Member, IEEE;* [**] *Senior Member, IEEE*

Abstract—This letter presents a neural architecture search to optimize tactile elaboration systems taking into account the computational cost of the whole pipeline consisting of data pre-processing and a convolutional neural network (CNN) model to extract information. The strategy is exploited to train standard 1D CNNs and binary CNNs on a 3-class touch modality classification dataset. The experimental results show that systems based on standard CNNs outperform state-of-the-art techniques in terms of accuracy and computational cost, while the ones based on binary CNNs reduce further the computational cost with a small accuracy drop.

Index Terms—CNNs, NAS, Tactile systems, Touch Modality Classification, Smart Sensors.

## I. INTRODUCTION

Modern prostheses equip tactile sensors to convey the sense of touch to humans. Effective and efficient wearable elaboration devices are required to collect and process the data from such systems. This letter addresses a touch modality classification problem [1] adopting an evolutionary neural architecture search (ENAS) [2] to design the whole elaboration pipeline consisting of the data preprocessing and the classification stages. The ENAS evaluates a custom loss function suitable for resource-constrained devices that, differently from standard approaches, takes into account the computational cost of both stages and the accuracy of the classifier because pre-processing could primarily affect the computational cost of the system when designing tiny classifiers. As a result of the ENAS evaluating the loss function proposed by [3], we outperform the state-of-the-art (SoA) accuracy and computational cost by adopting 1D convolutional neural networks (1D-CNNs) as classifiers. As a major result, the NAS enables using binary weights CNNs leading to half of the computational cost with a slight deterioration of the classification accuracy.

Touch modality classification is a well-known problem [4]–[8]. In [1], the authors performed three touch modalities on a piezoelectric sensing patch and they applied tensor-SVM and tensor-RLS algorithms to classify the data. In subsequent years, researchers proposed solutions to increase classification accuracy and reduce the computational cost: k-NN and SVM to address a two-class classification [9], [10], applying approximate computing techniques to deploy the algorithms on a resource-constrained device; transfer learning technique, transforming the data into RGB images [11]; recurrent neural networks (RNNs) to improve the accuracy and reduce the computational cost [12]; shallow CNNs that achieved a good trade-off between accuracy and computational cost [13], [14]; a kernel-SVM based on a reduced space that attained 85.4% accuracy at the expense of a huge computational cost [15]. All these works lack exhaustive search in the hypothesis space of the classifier architectures and/or on the pre-processing techniques applied to the data that affect the accuracy and the computational cost of the classification. Moreover, only a few tackled the hardware implementation on resource-constrained devices targeting shallow models.

NAS sets the SoA for designing tiny networks [16], [17] thanks to the capability of encoding hardware constraints directly in the procedure [18], [19]. In this letter, we merge the idea of taking into account the computational cost of the whole elaboration pipeline [3] with evolutionary NAS by evaluating a custom loss function to optimize the architecture. To the best of the authors' knowledge, no previous works based on NAS took into account the constraints on data preprocessing besides the DNN architecture. As a major result, the proposed procedure offers an automatic strategy to optimize a data elaboration system balancing the accuracy of the CNN classifiers and the computational cost of the whole pipeline.

The approach is not tied to specific implementation details or low-level optimization techniques. The deployment of the system on a device is influenced by the characteristics of the target hardware and the elaboration pipeline. Among the proposed elaboration steps, CNNs have the largest computational requirement. However, many optimized implementations for both specialized or general-purpose computing units support the atomic operations involved in the forward phase [19]–[21]. These optimizations can be applied to almost all CNN architectures yielded by our proposed procedure. Therefore, one can choose a target platform, retrieve hardware constraints, obtain the most effective combination of processing and CNN by adopting our strategy, and then deploy the architecture.

Experiments with a real-world dataset with 3 touch modalities sensed by an e-skin confirmed the suitability of the proposed procedure that based on the settings can generate very accurate systems, beating previous SoA results of more than 3% [13], [15], or very efficient systems based on binary CNN that significantly reduce the computational cost.

To summarize, i) we propose an ENAS for the exploration of the hyperparameters space of CNNs and the input data preprocessing techniques, addressing the touch modalities classification problem; ii) the results show that the ENAS provides a valuable strategy to optimize the hyperparameters of the architectures and the elaboration techniques applied to the input data, even setting a hard constraint on the computational cost.

Corresponding author: C. Gianoglio (e-mail: christian.gianoglio@unige.it).
The authors acknowledge partial financial support from AI-Powered Manipulation System for Advanced Robotic Service, Manufacturing and Prosthetics (Intelli-Man) project: EU H2022, Grant agreement ID 101070136.

Associate Editor: XXXXX XXXXXX.
Digital Object Identifier XX.XXXX/LSENS.XXXX.XXXXXXX

## II. METHODOLOGY

A recent work [3] proposed to optimize the elaboration pipeline using a loss function composed of two terms measuring generalization performance and computing the cost of the whole pipeline, respectively, obtaining SoA results for touch modality classification. The optimization problem can be formalized as:

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

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

where $\hat{L}_m$ is the empirical risk computed on the validation set $D_m$, $\theta$ is a hyperparameter that weights the computational cost of the processing pipeline $R_H$, $\mathcal{F}_i$ represents the space of the algorithms described by different values of hyperparameters, $\hat{L}_n(f)$ is the empirical risk computed on the training set $D_n$, and $\lambda$ weights the regularization term $R$ (e.g. L2-norm) that prevents overfitting.

In this letter, an ENAS [2] is enhanced pursuing the same result of [3]. The proposed approach optimizes the hyperparameters of a CNN and the data processing techniques simultaneously, taking into consideration the computational cost of the whole elaboration pipeline and the generalization performance of the architecture. The optimization procedure is supported by the ENAS which, iteratively, mutates parent models according to a search space generating an offspring. The offspring are evaluated and ranked accordingly to a loss function. The search is adjusted on the ranking to obtain a new pool of parents for the next iteration. The optimization ends when a stop criterion is met. The following sections detail our enhanced ENAS for touch modality classification, describing the search space, search algorithm, and evaluation criteria.

### A. Search Space

In this proposal, ENAS enables the tuning of hyperparameters of DNNs and processing techniques while considering the computational cost of the system. This allows for achieving a suitable balance between classification accuracy and computational cost. We employed standard CNNs and two variants based on the binarization of weights and the activation functions to classify the tactile data. In many applications [21], binary CNNs achieved accuracies similar to standard ones reducing the computational cost.

In detail, the three kinds of CNN adopted in the experiments are a standard 1D-CNN (1D), a 1D binary-weights CNN (BW) where the weights are forced to be -1 or +1, and a 1D full-binary CNN (FB) where both weights and the output of activations of the convolutional layers are binarized as -1 or +1. The 1D consists of blocks connected sequentially made of convolution, dropout, and average pooling (AP), resulting in a single-branch network. The last layer provides the classification label and consists of a convolutional layer with a number of filters equal to the number of classes and kernel size equal to 1, a global AP layer to reduce the size of each filter to one, and the Softmax layer. The BW and FB contain the same functional blocks of 1D, with a batch normalization layer after each AP layer. As hyperparameters of the CNN architectures which form the search space $\mathcal{SM}$ of the models, we chose the number of filters and kernel sizes of the convolutional layers.

As described in [3], a tactile sensing elaboration system (ES) consists of the sensing array, a pre-processing stage, and the inference stage. A datum $\mathcal{X} \in \mathbb{R}^{D_1 \times D_2 \times N}$ is collected by the sensing array, where $D_1 \times D_2$ is the geometry of the sensing patch ($D_2 = 1$ if the sensors can be represented as an array) and $N$ is the number of samples collected from each sensor. The pre-processing stage filters the data, reducing the noise and the number of samples. As a result, the $\mathcal{X} \longrightarrow \tilde{\mathcal{X}} \in \mathbb{R}^{D_1 \times D_2 \times \tilde{N}}$, where $\tilde{N} \leq N$. The resulting tensor $\tilde{\mathcal{X}}$ feeds a CNN providing the classification label. The data pre-processing affects both the accuracy and the computational cost of the whole pipeline [3]. Thus, besides the hyperparameters of the classifiers, the pre-processing techniques must be considered in the search space of the ENAS. In this work, besides not applying any technique to the raw data, we adopted similar processing previously used in [3] based on filtering: 1) a low-pass FIR filter with the hamming window, 2) a gaussian window convolved with the signals, and 3) a decimation technique to reduce the sampling frequency. A moving average with 50% of overlapped samples was also applied to reduce the number of data samples to three different values. The search space of the pre-processing techniques will be named $\mathcal{SP}$ in the following. It contains all the combinations of filtering and no-filtering with moving averages for an amount of 12 techniques.

### B. Search Algorithm

Procedure 1 depicts the ENAS search procedure. The ENAS initially generates a parent model $\mathcal{P}$ from the search space $\mathcal{SM}$. $\mathcal{P}$ is then trained with data processed by the technique extracted from $\mathcal{SP}$, solving (2) with a fixed $f = \mathcal{P}$. The ENAS computes the score by evaluating the loss function solving (1) with fixed $i$ and $\theta$. $D_n$ and $D_m$, in (1) and (2), correspond to the training and validation sets extracted from the processed data. At each step of the iterative procedure, the ENAS generates a child $C$ by applying two mutations to the convolutional layers of the parent architecture blocks: 1) either adding one block to the network (only if the maximum number of blocks is not achieved), either removing one block from the network (only if the minimum number of blocks is not attained), or no modification to the architecture; 2) a random mutation is always applied to a convolutional layer of a random block accordingly to $\mathcal{SM}$. As a result, the search on models adopts a schema based on blocks of single-branch architectures. The weight sharing technique [22] is also applied to enhance the accuracy performance resulting also in a faster search. After the training, the ENAS computes the score of $C$. The iterative procedure is repeated until a stop criterion is satisfied: either ENAS reaches the maximum number of epochs or the ENAS satisfies the early-stop criterion on the number of times none of the children achieved a better score than the parent model.

### C. Evaluation Criterion

Three constraints lead to the deployment of tactile sensing: inference time (IT), memory occupation (MO), and energy consumption (EC). On resource-constrained devices with limited parallelism, IT is proportional to the FLOPs number (simply FLOPs from now on) that must be run from the CPU. The memory is divided into two parts: flash memory hosting code and network parameters, and RAM storing partial results as tensors. The bottleneck is usually the RAM size, lower than the flash memory, since the tensors propagated through the system easily become large. EC is strictly correlated to the clock frequency, the total amount of operations, and the number of operations that the processor can execute in one cycle. Since the last performance highly depends on the targeted hardware and the

**Procedure 1** ENAS Search Procedure

0. **Input** Search spaces $\mathcal{SM}$ and $\mathcal{SP}$, User-defined parameter $\theta$ [3]
1. **Beginning Procedure**
   1) Generate a parent model $\mathcal{P}$ from $\mathcal{SM}$
   2) Train $\mathcal{P}$ with data processed by a technique picked from $\mathcal{SP}$
   3) save the score $score\_best$ computed by the loss function
2. **Iterative Procedure**
   1: **while** stop criterion is not satisfied **do**
   2: Apply a mutation to $\mathcal{P}$ based on $\mathcal{SM}$ generating a child $C$
   3: Train $C$ with data processed by a technique picked from $\mathcal{SP}$
   4: Compute the $score$ with the loss function [3]
   5: **If** $score > score_{best}$ **then** $\mathcal{P} = C$ and save the pre-processing technique as best **else** keep $\mathcal{P}$ and previous technique as bests
   6: **end while**
3. **Output** Return the best model and pre-processing technique

approach proposed in this paper is not designed for a specific device, this work proposes the evaluation of the loss function by measuring the computational cost $RH$ (1) as FLOPs or RAM MO during the ENAS search. In the first case, ENAS computes FLOPs for both the pre-processing and the classification stages; in the second, the ENAS computes the largest MO measured as the number of elements that have to be loaded in the RAM or cache memory of the resource-constrained device during the online inference. As a result, the largest MO corresponds to the size of the biggest tensor processed by the ES because the operations on the tensors are computed by the networks' layers sequentially, thus the output of a layer is saved into the RAM to be processed by the consequent layer. The FLOPs are computed accordingly to the convention presented in [3]. As an example, a MAC operation between two floating point (FP) numbers requires 2 FLOPs, one for multiplication and one for the summation with the cumulative result. In the case of binary networks, a MAC operation between an FP number and a binary weight requires only one FLOP since, when the number is multiplied by a negative weight, it just changes the sign thus only the summation matters. In the following, $\mathcal{L}_F$ and $\mathcal{L}_M$ will refer to the FLOPs and memory loss functions, respectively. During the evaluation of the two losses in the search procedure, the measured FLOPs and MO are normalized between 0 and 1 by their maximum values that can be computed a priori.

## III. RESULTS AND DISCUSSION

The dataset, available at https://github.com/cosmiclabunige/Touch_modalities_dataset, consists of three actions (i.e., slide a finger, roll a washer and brush a paintbrush) on a $4 \times 4$ sensing patch. Each action counts 280 data with a duration of 10 seconds sampled at $3KSamples/s$. Formally, $\mathcal{D} = \{(\mathcal{X}, y)_i; \mathcal{X}_i \in \mathbb{R}^{16\times 30000}; y \in \{Slide, Brush, Roll\}; i = 1, ..., 840\}$. $\mathcal{D}$ can be processed during the ENAS with 12 techniques, resulting from the combination of filtering proposed in [3] and moving average that reduces the number of samples to 50, 75, and 100. The pool of candidates networks hyperparameters is $filters = [4, 8, 12, 16, 32, 64]$ and $kernel\_size = [3, 4, 5, 6, 7, 8, 10, 12, 16]$. During the procedure data are split into training (480 data, 160 per class), validation (120 data, 40 per class) to evaluate the loss function, and testing (240 data, 80 per class) to compute the generalization performance. We set the minimum and maximum number of models' blocks to 2 and 7, the first parent model to 4 blocks, the dropout percentage value to 0.2, the pooling size to 2, the stride for convolutions to 1, a $learning\_rate = 1e-3$. The best model $\mathcal{P}$ is fine-tuned for 100 epochs with an early-stop criterion with a patience value of 8, and user-defined parameter $\theta = 0, 1, 5$, the maximum number of ENAS epochs $ste = 30$, the maximum number of iterations $max\_iter = 10$ defining the early-stop criterion mentioned in II-B, and the ENAS was run 5 times for each combination of model-$\theta$ and the results averaged. In the following, we first present the results of the accuracies attained by the ESs using the two loss functions on each model, based on the $\theta$ values. Next, we compare the accuracies, FLOPs, and memory usage of the ESs for the two loss functions on each model, again based on the $\theta$ values.

Table 1 presents the accuracy of the ESs based on the three models (1D, BW, and FB), evaluated on the test set and averaged on the 5 runs. The first column lists ESs, and the others show the average accuracy and the standard deviation for each ES evaluating the two loss functions. The table highlights in bold the accuracies that outperform the SoA results (85.4% in [15]). 1Ds outperform the other ESs' accuracies. Five out of six 1Ds present an accuracy higher than SoA. For 1Ds and BWs, when adopting $\mathcal{L}_M$, at the same value of $\theta$, the accuracies are higher with respect to $\mathcal{L}_F$. The BWs, with $\theta = 0$, present an accuracy drop lower than 3% with respect to the 1Ds and a slight improvement with respect to the SoA. When $\theta$ increases the accuracy drop widens up to ~6%. At an equal value of $\theta$, the accuracies achieved by BWs trained with $\mathcal{L}_M$ are slightly better than the ones obtained with $\mathcal{L}_F$. The FBs achieved the lowest accuracies with respect to the other networks, with a drop even higher than 20%. This deterioration is probably due to the hard approximations of the full-binary architectures.

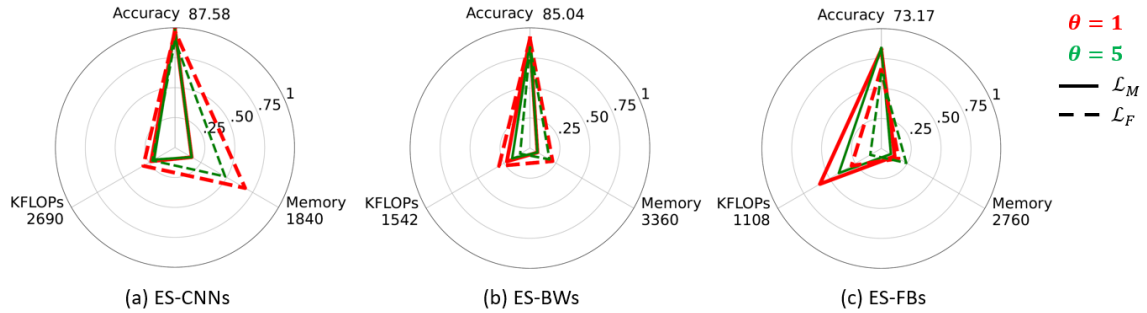
The radar plots in figure 1 show the ESs' accuracy, FLOPs, and MO with respect to $\theta$ values and the losses. Each plot displays a colored triangle for $\theta = 1$ and $\theta = 5$ where the vertices are the average accuracy, KFLOPs, and MO on the 5 runs. The values of the three performance were normalized with respect to the maximum values obtained with $\theta = 0$, represented by the numbers below each performance label. Since with $\theta = 0$ the ES computational cost is not relevant, the maximum accuracies in the figure result as the averages between the $\theta = 0$ values showed in Table 1. Figure 1(a) shows that FLOPs and MO of the 1Ds evaluated with $\mathcal{L}_M$ are lower than the ones attained with $\mathcal{L}_F$, at equal $\theta$ value. The reason is that constraining the MO, i.e. the dimensions of the tensors propagated through the systems, affects the FLOPs since smaller models are targeted by the ENAS. When $\theta = 0$, the 1Ds require ~2.69MFLOPs and 1840 elements in the memory; while, considering $\mathcal{L}_M$, the system takes ~695KFLOPs and 800 elements when $\theta = 1$, and ~621KFLOPs and 800 elements when $\theta = 5$. As a comparison with SoA, in [13] the ES achieved an average accuracy of 85% with ~1.31MFLOPs, thus the 1Ds with $\theta > 0$ present greater performance for both accuracy

Table 1: Accuracy results

| | **Memory Loss Function** | | | **FLOPs Loss Function** | | |
|---|---|---|---|---|---|---|
| **ESs** | $\theta = 0$ | $\theta = 1$ | $\theta = 5$ | $\theta = 0$ | $\theta = 1$ | $\theta = 5$ |
| **1D** | **87.58** ±1.00 | **87.08** ±1.39 | **86.83** ±1.31 | **87.58** ±1.22 | **86.50** ±2.70 | 83.58 ±1.64 |
| **BW** | **85.42** ±1.67 | 82.00 ±2.22 | 80.67 ±1.64 | 84.67 ±1.91 | 81.83 ±2.60 | 79.67 ±2.03 |
| **FB** | 72.67 ±3.71 | 69.08 ±5.32 | 69.42 ±4.72 | 73.92 ±1.91 | 65.50 ±1.89 | 64.58 ±5.37 |

This article has been accepted for publication in IEEE Sensors Letters. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/LSENS.2023.3273733






(a) ES-CNNs (b) ES-BWs (c) ES-FBs

Fig. 1: Radar plots of systems performance. The solid lines refer to $\mathcal{L}_M$, while the dashed to $\mathcal{L}_F$. Red lines represent $\theta = 1$ and greens $\theta = 5$.

and computational cost measured as FLOPs. Figure 1(b) shows that, considering $\mathcal{L}_F$, FLOPs and MO of BW with $\theta = 1$ are the highest. When $\theta = 5$, BW with $\mathcal{L}_F$ attains the lowest FLOPs but with a similar value of MO of $\theta = 1$. On the other hand, the BWs with $\mathcal{L}_M$ are better in terms of MO and intermediate results in terms of FLOPs. When $\theta = 0$, the BWs require ~1.54MFLOPs and 3360 elements in the memory; considering $\mathcal{L}_M$, the system takes ~421KFLOPs and 800 elements when $\theta = 1$, and ~347KFLOPs and 800 elements when $\theta = 5$; while, with $\mathcal{L}_F$, the system takes ~530KFLOPs and 1200 elements when $\theta = 1$, and ~236KFLOPs and 1120 elements when $\theta = 5$. Regarding 1Ds, the BWs require much lower FLOPs; while concerning the MO the BWs present a bigger tensor when $\theta = 0$, while it has similar sizes when $\theta > 0$. Hence, besides the drop in accuracy, BWs are valuable options when the computational cost measured as FLOPs is relevant. Eventually, looking at Fig. 1(c), the FBs fail to improve in terms of FLOPs and MO with respect to the BWs, except in the case of $\theta = 0$. In any case, the high drop in accuracy of the FBs makes them unsuitable for an embedded implementation.

Summarizing, the 1Ds achieved the best results in terms of accuracy also outperforming the SoA, 85% in our previous work [13] and 85.4% in [15], for both the loss functions and for all the $\theta$ values but $\theta = 5$ with $\mathcal{L}_F$. Moreover, with $\theta > 0$, the FLOPs of 1Ds are lower than 700K with respect to ~1.31MFLOPs in [13]. Eventually, BWs are valuable options when the computational in terms of FLOPs is the hardest constraint.

## IV. CONCLUSION

The paper proposed an enhanced ENAS to optimize a tactile elaboration system for touch modalities classification. The ENAS, in the search procedure, evaluates the computational cost of the pre-processing and classification stages using a customized loss function. Results show that standard 1D-CNNs attain the best accuracy, outperforming the SoA by more than 3% while binary weights CNNs present a better computational cost than standard 1D-CNNs, with a drop in accuracy of at most 6%.

## REFERENCES

[1] 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.

[2] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," IEEE Transactions on Neural Networks and Learning Systems, pp. 1–21, 2021.

[3] C. Gianoglio, E. Ragusa, P. Gastaldo, and M. Valle, "A novel learning strategy for the trade-off between accuracy and computational cost: A touch modalities classification case study," IEEE Sensors Journal, vol. 22, no. 1, pp. 659–670, 2021.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] M. Alameh, Y. Abbass, A. Ibrahim, G. Moser, and M. Valle, "Touch modality classification using recurrent neural networks," IEEE Sensors Journal, vol. 21, no. 8, pp. 9983–9993, 2021.

[13] C. Gianoglio, E. Ragusa, R. Zunino, and M. Valle, "1-d convolutional neural networks for touch modalities classification," in 2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS). IEEE, 2021, pp. 1–6.

[14] F. Sakr, H. Younes, J. Doyle, F. Bellotti, A. De Gloria, and R. Berta, "A tiny cnn for embedded electronic skin systems," in International Conference on System-Integrated Intelligence. Springer, 2023, pp. 564–573.

[15] 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, vol. 19, no. 2, pp. 959–968, 2021.

[16] D. Baymurzina, E. Golikov, and M. Burtsev, "A review of neural architecture search," Neurocomputing, 2021.

[17] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," ACM Computing Surveys (CSUR), vol. 54, no. 4, pp. 1–34, 2021.

[18] H. Benmeziane, K. E. Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "A comprehensive survey on hardware-aware neural architecture search," arXiv preprint arXiv:2101.09336, 2021.

[19] L. Sekanina, "Neural architecture search and hardware accelerator co-search: A survey," IEEE Access, vol. 9, pp. 151 337–151 362, 2021.

[20] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," Neurocomputing, vol. 461, pp. 370–403, 2021.

[21] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, "Binary neural networks: A survey," Pattern Recognition, vol. 105, p. 107281, 2020.

[22] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," arXiv preprint arXiv:1806.09055, 2018.