# Compensation of Cross-talk in a Large Piezoresistive Sensor Array

Carlos Garces Maldonado
*dept. Data Management and Biometrics Group*
*Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS), University of Twente*
Enschede, Netherlands

***Abstract*—Flexible resistive sensors have applications in modern devices that implement tactile sensors for pressure and force sensing. The e-Cone is a low-cost prototype device developed by the University of Twente to measure grip strength. The last version of this sensor has a high resolution of 64x128 sensor pixels. However, the sensor exhibits significant cross-talk, which adversely affects the readout of sensing pixels when multiple pixels are simultaneously pressed. To address this issue, we present an evaluation method that utilises spring mechanisms to study the behaviour of a single sensing pixel. Additionally, we propose a modification to the acquisition PCB circuit aimed at reducing the cross-talk. Finally, we introduce a machine learning approach to compensate for the cross-talk effect, thereby enabling reliable pressure pattern measurement.**

***Index Terms*—Velostat, pressure sensor array, cross-talk, sensing pixel, machine learning.**

## I. Introduction

Piezoresistive sensor arrays (PRS) have numerous applications in consumer electronics, telepresence and soft robotics, automation, and healthcare monitoring devices to measure force or pressure [1]. These sensors work with the piezoresistivity principle; the electrical resistance of the semiconductor or polymer changes due the mechanical strain. By employing flexible materials, piezoresistive sensors can achieve flexibility, enabling a wide range of different applications, including wearable electronics, which surpasses some of the limitations of rigid materials [2]. One of the cheapest alternatives for building a flexible PRS is to use Velostat material. The Velostat is a polymer material filled with carbon particles to make it able to be electrically conductive. Figure 1 illustrates the principle of the sensor: as pressure is applied to the material, the proximity of carbon particles increases, resulting in improved conductivity and a reduction in resistance for the current flow [3].

One of the main challenges in large PRS arrays is the cross-talk effect. This phenomenon occurs when the sensing elements (or sensing pixels) are in close proximity to each other. The signal from one sensor pixel influences the other adjacent elements leading to inaccurate readings. The cross-talk effect is due to alternative current paths created that are read out by the acquisition system [4]. In order to mitigate the cross-talk effect, complex circuits can be designed, or as demonstrated in this work, computational compensation approaches can be implemented.

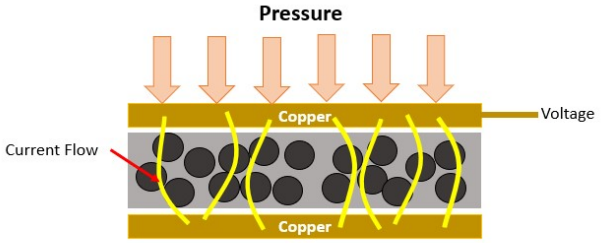



Fig. 1: Sensor construction principle

The objective of this research is to develop a novel approach to develop a compensation method for mitigating cross-talk in a large resolution PRS array. The main research question is: Is there a computational compensation method suitable for compensating cross-talk in this type of sensor? In addition to the main question, the research aims to answer the following sub-questions:

- What is an appropriate experimental setup for collecting data to describe the behaviour of the sensor array, including the cross-talk effect?
- What computational methods can be used to compensate for the cross-talk effect in the sensor array?
- What is the nature of cross-talk and its behaviour from a quantitative perspective?

This work begins with Section II, which explores related studies on addressing the cross-talk effect. Section III provides an overview of the background related to the development of the e-Cone sensor. In Section IV, the system used to evaluate the sensor and generate data for the computational approach is described. Section V focuses on studying the behaviour of a single sensing pixel's response in cross-talk scenarios. Section VI presents the computational compensation approach employed. Lastly, Section VII gives a detailed discussion, while Section VIII presents the conclusion.

## II. Related Work

The cross-talk issue can be addressed and examined from three different perspectives. Firstly, the mechanical perspective requires detailed considerations of the sensor's design and construction to avoid undesired interactions between sensing points. Secondly, the electrical perspective acknowledges the

circuit properties that possibly influence the creation of alternative current paths that affect the sensing pixel measurement. Lastly, the computational perspective explores the utilisation of algorithms and software tools for compensating the affected measurements.

*1) Mechanical design to reduce cross-talk:* These approaches involve structural changes to the transduction mechanism. When pressure is applied to the sensor, the resistance of each element changes in proportion to the applied pressure. However, when two sensing pixels are placed so close to each other, the adjacent element is also pressed. Mechanical methods focus on increasing the distance between sensing pixels and isolating them. Harris [5] shows how an accurate design of the sensor's dimensions can play a major role in how the measurement can be affected. Another example is introducing a spacer layer between the electrodes and the piezoresistive element (Velostat) to reduce cross-talk [6]. The main drawback of this solution is that it reduces the sensors' sensitivity.

*2) Electrical cross-talk reduction techniques:* The electrical perspective approaches involve increasing the complexity of acquisition circuits. One option is to add diodes to avoid the flow of current where alternative current paths can be created [2], although this can be effective, this involves a complex manufacturing process in high-resolution sensors. Another option is arrangements with the operational amplifiers in such a way that the unwanted current paths are short circuits forcing them to a zero potential. The Voltage Feedback Method (VFD) is one of those techniques, which applies a feedback voltage to the sensing elements. By adjusting the feedback voltage, the cross-talk between adjacent sensing elements can be reduced. Although this method can be effective, it involves complexity in the circuit, and the feedback current needs to be high depending on the ratio of the resistances [7].

An alternative technique is called the Zero Potential Method (ZPM), where all row or column nodes are placed at virtually equal potential zero using the virtual ground of high-gain operational amplifiers with a negative feedback configuration. ZPM and VFM both look momentarily to virtual short-circuit around non-scanned sensels. Usually ZPMs circuits have better results in large sensor array dimensions [8]. These methods have the downside of raising the cost of sensors and increasing the number of components used on them, which may be a disadvantage in situations where there are space constraints and expanding the size of the board is not feasible.

*3) Computational cross-talk compensation methods:* In recent years, there has been a growing interest in applying neural networks to solve problems that are difficult to model mathematically, such as the case of the cross-talk in large PRS arrays. Neural networks can effectively learn complex relationships between sensor measurements and generate certain compensation for the cross-talk issue. Nevertheless, most of the work is related to pattern recognition rather than precise pressure or force value estimation, which are the cases of [4] and [9]. They propose using convolutional neural networks to correct cross-talk effect without using sophisticated hardware.

In the case of Müller's work [4], they use an analytical forward model to find the relation between sensing pixels' resistance and measured ADC voltages considering alternative current paths in order to generate synthetic data for training the network. Around 100,000 random 16x16 patterns are drawn using the analytical forward model. For correcting the measurement error captured in the matrix of ADC-measured voltages, they use the approximation capabilities of neural networks for mapping from ADC voltages to the pressure matrix.

Unlike Müller's work, our analytical forward model is different due to the distinct nature of our acquisition circuit. Our compensation method is tailored to a high-resolution sensor (128x64 sensing pixels) instead of a 16x16 one. Additionally, our machine learning approach is trained using experimental data rather than synthetic data to avoid finding a complex analytical forward model. Furthermore, we adopt a sequential compensation technique, accurately estimating the load value for each individual sensing pixel. In Müller's work, the pressure accuracy is lower and just sufficient for classification and object recognition, while in this work, the objective is to have accurate measurements for medical diagnosis.

## III. Background

Grip strength is an important indicator of physical health and a variable to evaluate disability, morbidity, and mortality. [10]. To measure grip force, different devices have been developed over time; hand dynamometers (hydraulic, mechanical, or digital), these are handheld devices providing a single value that represents overall grip strength. Pinch gauges measure the force between the thumb and fingers and they are used to assess hand functioning. And sphygmomanometers, which are blood pressure cuffs, are used to measure grip strength as well [11]. The previous devices offer a limited range of measurements for hand muscle behaviour, often providing only a single value that generalises a patient's grip strength. For that reason, over the previous years, the instrument called e-Cone has been developed at the University of Twente. This sensor consists of a PRS array wrapped around a cone (or cylinder in the most recent version as seen in Figure 2) that allows visualising the grip strength pattern. Figure 3 illustrates a hand grip pattern taken by the sensor.

### A. Sensor Hardware

The sensor consists of a matrix of 8192 elements in a configuration of 128 columns and 64 rows. It uses a Velostat sheet between two electrodes which has a surface resistance of 31,000 $\Omega$ and a volume resistance of 500 $\Omega \cdot cm$. The electrodes are flexible printed circuit boards (PCBs) with multiplexers soldered to them. The sensor data is read out by an acquisition PCB board that communicates to a Raspberry Pi Zero W (Figure 4). The Raspberry Pi processes the data and does wireless data transmission to client devices. The e-Cone structure is 3D printed, consisting of a cylinder that makes it easier to mount the PRS array compared to mounting the array to a cone surface.

Fig. 2: e-Cone most recent version

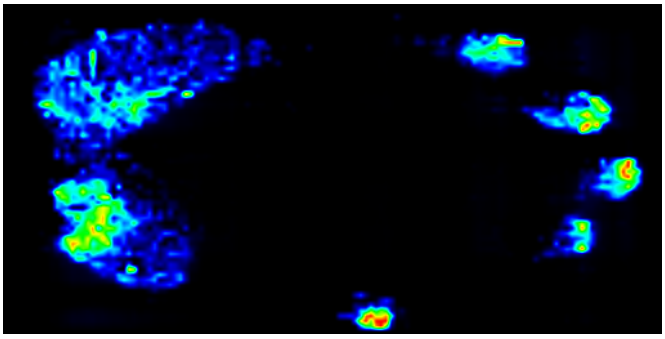


Fig. 3: Sensor readout; hand pattern visulatization

### B. Pressure Sensing Principle

The acquisition board reads the voltage of each sensing pixel in a sequential manner. Figure 5 is a simplification of the e-Cone circuit using 2x2 sensing pixels version to show the concept. It illustrates the sensing principle and the basic elements to do it; the piezoresistive material (Velostat), the amplifier, the power supply for $V_{test}$, the sensing pixels represented by resistances, and the multiplexers. The transparent paths are below the Velostat material while the strong-coloured paths are above (purple-colored area).

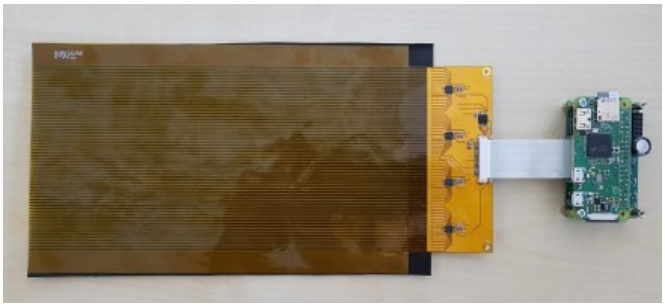


Fig. 4: Pressure sensor, acquisition board, and Raspberry Pi Zero W

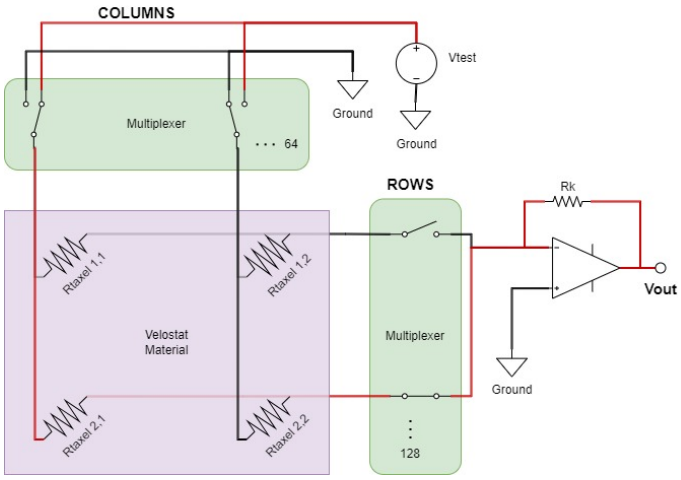



Fig. 5: Scanning circuit overview (simplified)

The sensor has an array of columns that are connected to a 16:2 channel analogue multiplexer. The multiplexer outputs to $V_{test}$ or ground depending on the selected control signal. Only one column can be activated to $V_{test}$ while the others go to ground (column deactivated). In the rows array, a 32:1 channel analogue multiplexer is used, which closes the circuit to send a current through $V_{test}$ to the Operational Amplifier (op-amp). To read out a certain sensing pixel, a specific column is activated, followed by a row. Then, the op-amp amplifies the signal with the relation of $R_k$ and the $R_{taxel}$ as is seen in Figure 6. The amplified voltage is read by the ADC integrated circuit located in the acquisition board. In the example of Figure 5, the red path is the closed circuit due to some load on the sensing pixel therefore, $V_{out}$ can be described by the next equation which is given by the op-amp configuration:

$$V_{\text{out}} = -V_{\text{test}} \frac{R_k}{R_s + \frac{R_k}{A} + \frac{R_s}{A}} \approx -V_{\text{test}} \frac{R_k}{R_s} \qquad (1)$$

Where $R_s = R_{taxel2,1}$ (the resistance of the measured sensing pixel), $A$ is the amplification factor of the amplifier. Typically A $>$ 100 000, allowing to approximate the function to a relation between $R_k$, $R_s$, and $V_{test}$.

### C. Cross-talk Model

Cross-talk, in the context of PRS array, refers to the appearance of extra unwanted current paths around the target sensing pixel ($R_{taxel}$ or $R_s$). The current flowing in these unwanted paths should be included in the $V_{out}$ calculation [12], otherwise the measured voltage of the sensing pixel is not accurate anymore. Previous work [13] describes the cross-talk effect based on the influence of the neighbouring activated sensor cells. The alternative paths' unwanted effect is modelled by a resistor $R_L$ from op-amp negative $(-)$ input to ground (as seen in Figure 6).

$R_L$ summarises all the equivalent resistance of all alternative created paths. The mathematical model considering this resistance into (1) is:

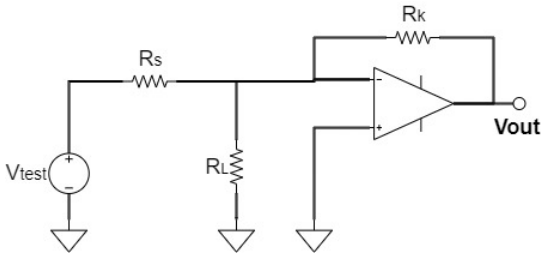
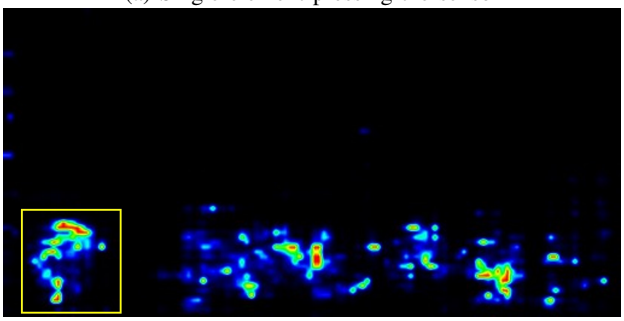



Fig. 6: Model considering the cross-talk effect in [13]

$$V_{\text{out}} = -V_{\text{test}} \frac{R_k}{R_s + \frac{R_k}{A} + \frac{R_s}{A} + \frac{R_s R_k}{A R_L}} \approx -V_{\text{test}} \frac{R_k}{R_s \left( \frac{R_k}{A R_L} + 1 \right)} \tag{2}$$

If there are not too many alternative paths then the $R_L$ factor can be neglectable since its value would be large enough to neglect $\frac{R_k}{AR_L}$. Nevertheless, this $R_L$ can affect if too many alternative paths are there or if the $R_L$ has a noticeable small value because it is pressed too hard. The visual effects of cross-talk can be seen in Figure 7 where the reference (yellow squared element in 7a) readout drops when more elements are pressed in the same row direction (7b).

(a) Single element pressing the sensor



(b) Multiple elements pressing the sensor

Fig. 7: Cross-talk effect visual imaging

### D. Previous Sensor Evaluation and Reliability

The assessment of the e-Cone sensor in the past was constrained by its qualitative validation, rather than being quantitatively evaluated in detail. A comprehensive investigation into the behaviour of an individual sensing pixel was lacking in the evaluation because the average of measured ADC voltages of a group of pixels was taken. In order to propose a computational method, it is crucial to comprehend the behaviour of individual sensing pixels and the interplay between them, as one sensing pixel may influence others, and vice versa. To facilitate sensor evaluation and data collection, we designed an experimental flexible setup that allows for the study of individual sensing pixels under cross-talk conditions when multiple sensing pixels are pressed.

## IV. Evaluation Method Development

The purpose of the sensor evaluation is to get a better understanding of the performance of the sensor array and to give a better description of the cross-talk effect in our sensor. The evaluation checks on the next parameters:

- Reliability and repeatability: sensor's ability to produce consistent output readings when it is subjected to the same pressure of force repeatedly.
- Sensitivity: how the output voltage changes in response to a change in pressure in function of $V_{test}$ parameter.
- Linearity: check if there is a proportional change in output with changes in input, or non-proportional change if the sensor works in a non-linear region.
- Cross-talk behaviour: how a single sensing pixel measurement changes when more sensing pixels are pressed.

### A. Experimental Setup

The experimental setup aims to test the previously mentioned features of the PRS array. To evaluate the sensor is critical to know the applied load in the sensing pixel and the area that is covering that load. The sensor is subjected to different levels of pressure set by a pressure mechanism. By obtaining information about the pressure value and the corresponding sensing pixel under evaluation ADC voltage measurement we can qualitatively asses the sensor response. The experimental setup consists of a table where the load can be placed in specific sensor regions (Figure 8). The table has the flexibility to place one or more loads in the same column or same row. Multiple pressure mechanisms can be utilised to generate diverse pressure patterns within the sensor.

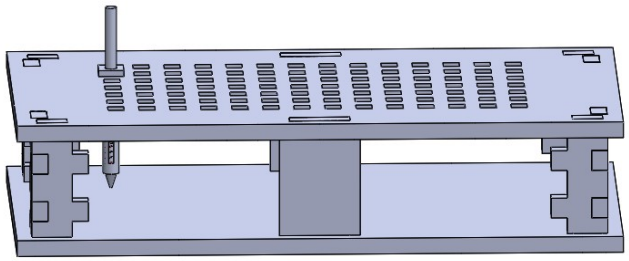


Fig. 8: Experimental setup for data collection (Lateral view)

The table columns and rows are labelled, as seen in Figure 9. The labels help to have better control of the experiments performed and easily register them in $.txt$ files (i.e. pinpoint pressure mechanisms placed on B7 and C7).

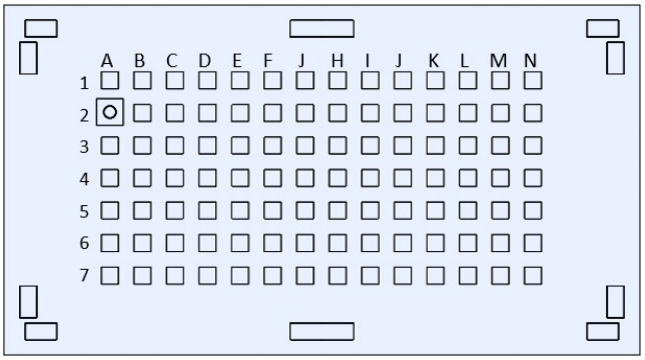



Fig. 9: Experimental setup for data collection (Top view)

### B. Pressure Mechanism Design

The pressure mechanism applies a load to the Velostat material, simulating the pressure range that the hand can apply when gripping an object. The mechanism consists of several components, including a tip that can be interchanged to adjust the area of pressure, a spring with a specific force per millimetre of compression, a tube that guides the spring in a linear fashion, and a pushing tube that compresses the spring (Figure 10).

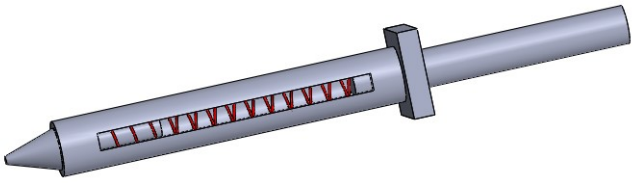


Fig. 10: Pressure mechanism system

*1) Spring Selection:* The spring is selected based on commercial availability and defining a range according to hand pressure variability. According to [14], [10] the maximum hand pressure a human can exert varies depending on physical condition, age, and gender. But, the average maximum hand pressure that a healthy adult can exert is around 0.4 MPa (megapascals) to 0.7 MPa. That means that the pressure mechanism should be able to apply that range of pressure to simulate a hand grip strength. To properly select the spring for the sensor evaluation, the pressure formula which relates force and area (3) is used to determine the appropriate spring constant.

$$P = F/A \tag{3}$$

The load weight applied by the pressure mechanism is experimentally validated using a high-resolution scale to ensure that the applied load is proportional to the compression distance of the spring. Figure 11 shows this linear relation.

| Contact Area ($mm^2$) | Max. Force ($N$) | Pressure ($MPa$) |
|---|---|---|
| 1 | 1.53 | 1.53 |
| 2 | 1.53 | 0.765 |
| 4 | 1.53 | 0.3825 |
| 8 | 1.53 | 0.1913 |

TABLE I: Max. pressure for a 0.06 $N/mm$ spring resolution

| Contact Area ($mm^2$) | Max. Force ($N$) | Pressure ($MPa$) |
|---|---|---|
| 1 | 2.57 | 2.57 |
| 2 | 2.57 | 1.285 |
| 4 | 2.57 | 0.6425 |
| 8 | 2.57 | 0.3213 |

TABLE II: Max. pressure for a 0.08 $N/mm$ spring resolution

| Contact Area ($mm^2$) | Max. Force ($N$) | Pressure ($MPa$) |
|---|---|---|
| 1 | 3.64 | 3.64 |
| 2 | 3.64 | 1.82 |
| 4 | 3.64 | 0.91 |
| 8 | 3.64 | 0.465 |

TABLE III: Max. pressure for a 0.09 $N/mm$ spring resolution

It is crucial to take into account an offset that includes the weight of the tip, push tube, and spring components, in order to accurately determine the total load applied to the sensing pixel. The total offset weight in the mechanism is 1.7 grams. Tables I, II, and III show pressure ranges depending on the selected spring and the 3D printed pinpoint area for the mechanism tip. The mechanism design has the flexibility to be modified to distribute pressure over a larger area; however, this may result in a reduced probability of uniform pressure distribution due to material roughness caused by the manufacturing process. In contrast to large pinpoint areas, small pinpoint areas are more difficult to manufacture and can cause load shifts when applying force in the sensor.

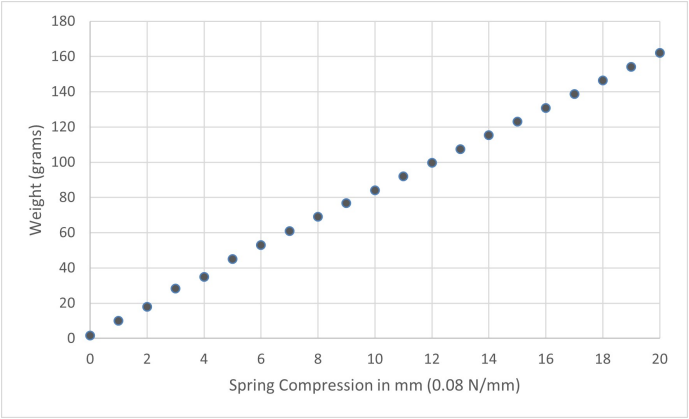



Fig. 11: Relation of a spring compression ($0.08N/mm$) and weight scale measurement

*2) Evaluation System Manufacturing:* The system setup is constructed using a combination of 3D printing and laser-cutting manufacturing processes. The system was designed

in CAD software, and the pressure mechanism, along with certain top-side table variations, were made using 3D printing technology. The 3D printing process enables achieving a precise and customised design which is crucial for simulating the range of pressure that the sensor array may encounter. Additionally, the laser-cutting process was used to cut the table that holds the pressure mechanism. This process allows for making an accurate and sturdy platform that can securely hold the pressure mechanisms during testing and data collection. Figure 12 shows one version of the experimental table setup.

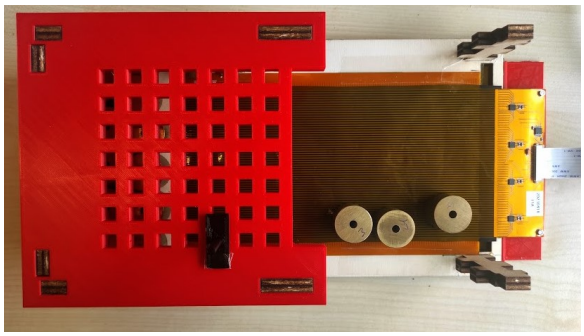
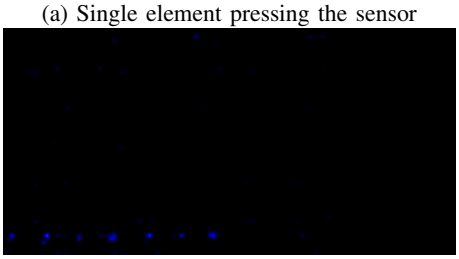


Fig. 12: Experimental test setup

### C. Data Collection

To collect data, the pressure mechanisms are used to apply loads to the sensor array while it is in operation; Figures 13a and 13b show the sensor readouts while performing the experiments. The sensor outputs are recorded in a $.txt$ file document which later can be interpreted and processed. To ensure the accuracy and reliability of the data, multiple trials for each pressure level are performed.

(a) Single element pressing the sensor



(b) Multiple mechanisms pressing the sensor

Fig. 13: Readout example for data collection

In order to generate different pressure patterns, 'steps' structures were 3D printed as seen in Figure 14 and 15. The steps in the structure are well known so the 'step levels' are linked to the spring linear compression. Additionally, the table setup is modified to cover half of the sensor, allowing for the placement of various loads such as circular weights, finger pressure, or the palm hand area of a hand. These loads are used to introduce disturbances (cross-talk) to the reference labelled pinpoint which is crucial for generating training data for the compensation method.

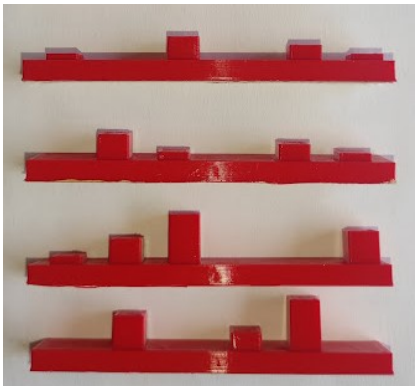


Fig. 14: Step structures with different level patterns

Fig. 15: Load patterns with multiple pressure mechanisms

Finally, Figure 16 illustrates the process of pressing the mechanism. The mechanism is manually pressed, and two scales are positioned behind it to avoid any slope between points A and B.

### D. Evaluation and Compensation Software

C++ language is used for the sensor's ADC readout utilising the Raspberry Pi, as it provides a fast and efficient way to gather data from the sensor. However, this language has its limitations when it came to data visualisation and ease of use for non-programmers. As a result, for the data processing and compensation method, we switched to a Python-based software solution. Python has a number of advantages for scientific data analysis, including libraries for data visualisation, processing, and machine learning (ML). For this application, a Python-based software solution is developed, which enables

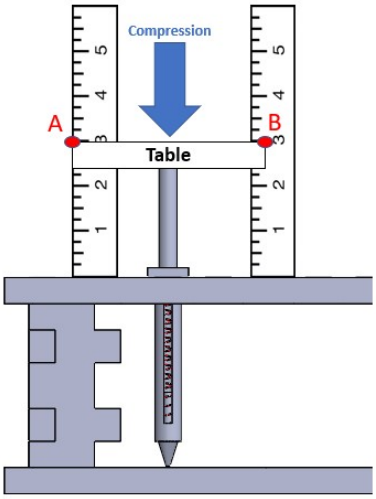



Fig. 16: Mechanism compression process for collecting data

the automatic filtering of behaviour data of any selected sensing pixel, array row, or column. The Python language is also utilised for developing the computational compensation method and training the ML model.

## V. Sensor Evaluation & Cross-talk Study

In the sensor evaluation, first reliability and repeatability are checked. Then, the method addresses and extends the previous mathematical model in order to understand better its performance characteristics. Finally, different experiments are performed to validate the mathematical model in a quantitative way and to observe how the voltage measurement of a sensing pixel drops when there is cross-talk.

### *A. Reliability and Repeatability*

While the sensor provides valuable data, its reliability may be limited. Inconsistencies are observed, as seen in Figure 17 at times during the readings.

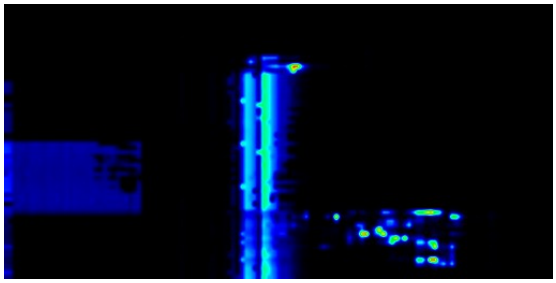
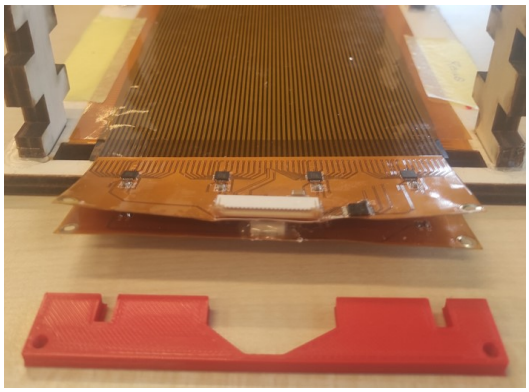
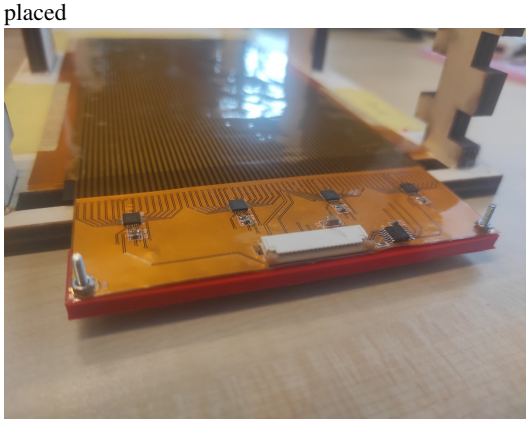


Fig. 17: Errors in readout due to connectivity issues

During the troubleshooting process, it was discovered that there were connectivity issues in the part of the flexible PCBs subsystem. Rigid Integrated Circuits (ICs) were possibly breaking their solder points to the PCB due to the stress put on them by bending the PCBs. The major issue was observed in the interconnection between the two flexible PCBs (columns and rows PCBs). The connector was not able to maintain a stable connection, likely due to a lack of proper support or constant bending in this area. To address the bending condition, a spacer was designed and placed between the two flexible PCBs to prevent them from bending excessively and to provide additional support and stability for the components (as seen in Figure 18). New flexible PCBs were soldered, requiring careful attention to detail and precision.



(a) Bending in the region where components are placed



(b) No bending in the components region.

Fig. 18: Spacer solution

To assess the measurement consistency, a total of 100 measurements were conducted over a period of time, both before and after implementing the connector fixing solution. The results of these measurements are shown in Figure 19 and Figure 20, respectively. The percentage of outliers in the measurements was then calculated as means of evaluating their consistency before and after the connectivity fixing. This is shown in Table IV, where outliers drastically drop after the fixing denoting the importance of ensuring good connectivity within the sensor. By conducting this straightforward evaluation, it can be certain of the quality of the measurements prior to utilising them as valid data for the compensation method and the effectiveness of the fixing solution. This assessment provides insight into whether a redesign iteration is necessary or not.

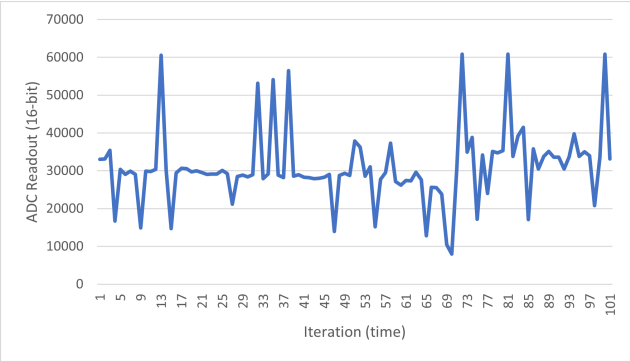



Fig. 19: Repeatability test before connectivity fixing

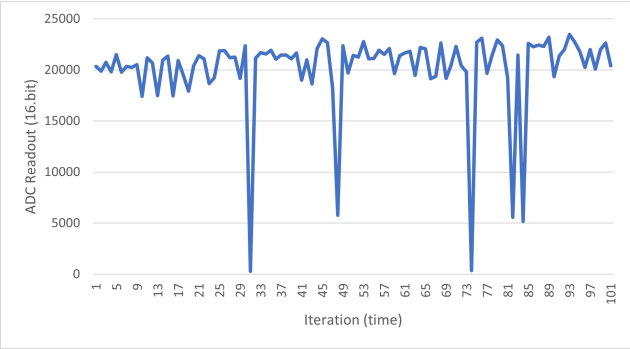



Fig. 20: Repeatability test after connectivity fixing

| Sensor | Outliers (%) |
|---|---|
| Flexible PCBs with non-fixed connections | 22% |
| Flexible PCBs with fixed connections | 5% |

TABLE IV: Percentage of outliers in data

### B. Theoretical Analysis

The previous mathematical model to describe and understand the cross-talk effect in the PRS array has some limitations in order to be demonstrated. Firstly, the model for the cross-talk was checked only in a qualitative way and not in a quantitative way, which can limit the understanding of the cross-talk effect. Secondly, the cause of cross-talk in the sensor is simplified as $R_L$ (equation 1) and not studied in detail as a function of a specific number of possible alternative current paths that could exist in the sensor when multiple sensing pixels are pressed in the sensor.

Figure 21 illustrates the possible current alternative paths when a readout of a sensing pixel takes place (green-coloured paths). These alternative paths happen due to sensing pixels pressed in the same row while the sensor is doing the readout of just one ($R_i$) and a path when sensor pixels are pressed in the row and column in such a way that it closes the circuit ($R_m$). To validate the equation 2, $R_m$ is currently ignored. This is because the previous work observed a significant drop when multiple loads were placed in the same row, rather than when multiple sensing pixels were activated in the same column.

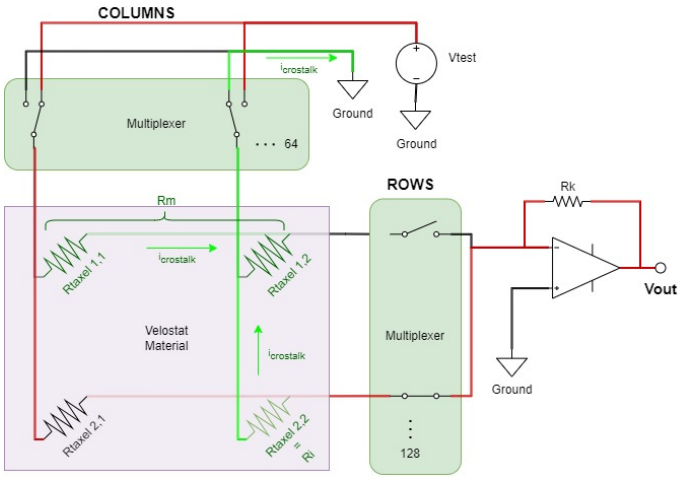



Fig. 21: Alternative current paths in a sensing pixel reading

Looking at the green-coloured path, then $R_L$ is extended as multiple resistances in parallel if more resistances $R_{taxel}$ are pressed over the same current line (red-coloured path), therefore extending the previous equation 1:

$$R_L = \left( \sum_{n=i}^{i} \frac{1}{R_i} \right)^{-1}$$

$$V_{\text{out}} \approx -V_{\text{test}} \frac{R_k}{(R_{taxel} + R_{mux}) \left( \frac{R_k}{A} (\sum_{n=i}^{i} \frac{1}{R_i}) + 1 \right)} \quad (4)$$

Where $R_i$ is the specific parasitic resistance due to pressing sensing pixel $n$, $i$ is the number of alternative current paths which depend on the number of sensing pixels pressed on the same row. Its value can go from 1 to 128 (the maximum of sensing pixels in the row). $A$ is the op-amp amplification factor. Considering $R_{mux}$ small by selecting a good multiplexer, the next equation simplification can be made:

$$V_{\text{out}} \approx -V_{\text{test}} \frac{R_k}{R_{taxel} \left( \frac{R_k}{A} (\sum_{n=i}^{i} \frac{1}{R_i}) + 1 \right)} \quad (5)$$

It is important to notice that the $R_i$ value depends on the pressure intensity on that region. The expansion of $R_i$ as the sum of different created paths allows understanding better the variables that play a role in the previous equivalent resistance $R_L$ (1) and therefore define these as feature inputs in a machine learning model. It is noticed by the model that $R_i$ reduces $V_{out}$, and a large value of $i$ can suppress any observation of $V_{out}$ when the sensing pixel $R_{taxel}$ is pressed. To validate the equation with experimental data, it is possible to simplify the model further. Assuming that the pressure or load is applied uniformly across the sensor (or multiple sensing pixels in the same row) by the mechanisms, the $R_i$ in eq. 5 can be considered equal in each element of the sum. Therefore:

$$\sum_{n=i}^{i} \frac{1}{R_i} = \frac{N}{R_i}$$

$$V_{\text{out}} \approx -V_{\text{test}} \frac{R_k}{R_{taxel}\left(\frac{NR_k}{AR_i}+1\right)}$$

$$V_{\text{out}} \approx -V_{\text{test}} \frac{AR_kR_i}{(AR_i+NR_k)\,R_{taxel}} \tag{6}$$

Equation 6 facilitates the assessment of the anticipated sensor response in accordance with the theoretical model in the conducted experiments. Therefore, evaluate if the sensor is giving the correct response or not, based on the model's predictions. Finally, an interactive model is done to manipulate the values and observe how the sensor response changes in relation to the model parameters. Figure 22 illustrates the interface of the interactive model. Up to 4 different model behaviours can be set and compared where the graph y-axis shows the $V_{out}$ sensing pixel response in function of how many sensing pixels are pressed in the same row ($N$ in x-axis).

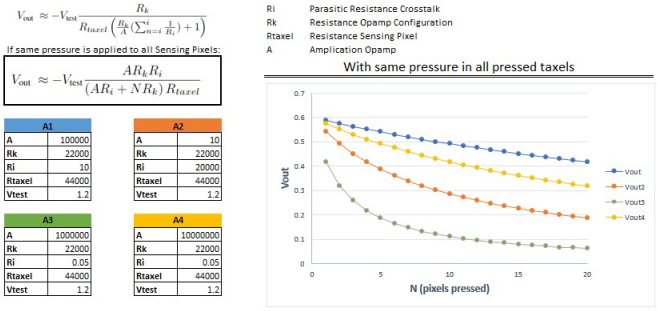



Fig. 22: Interactive model interface

## C. Model Validation

In order to validate the theoretical model, experimental data collection is done in a systematic way. The experimental data is used to compare predictions of the mathematical model with the actual behaviour of the system. This is done by plotting the model predictions alongside the experimental data and visually comparing them. Then, the results are analysed. If the model predictions match the experimental data closely, then the model can be considered valid. If there are discrepancies, then the model may need to be modified or refined. Collecting experimental data in this system is challenging due to various factors that can affect the accuracy and reliability of the data. The most common issue is illustrated in Figure 23, which shows the possible misalignments that can occur between the pressure mechanism and the studied sensing pixel.

The misalignment occurs due to mechanical drift while applying the pressure, human error during the setup of the experiments, and imprecision in applying the mechanisms in the sensor. When the mechanism and the sensing pixel are not aligned correctly, the $V_{out}$ recorded varies and may not accurately represent the behaviour of the system, leading to

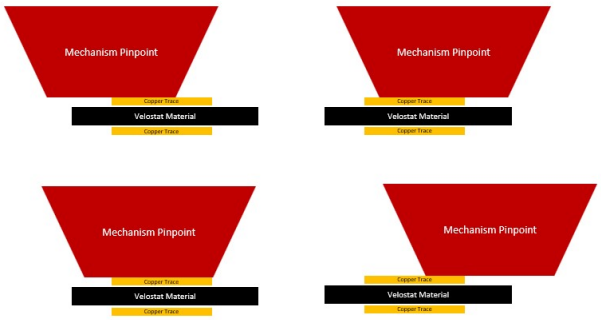



Fig. 23: Possible scenarios for pinpoint pressing sensor

errors or biases in the analysis. To address the misalignment issue, performing multiple replicates of an experiment is done. By conducting the experiment multiple times, any error will be spread across the replicates, reducing their overall impact on the results. The mean value of the replicates is calculated to cancel any random error and to show the underlying signal behaviour to become more apparent.

The relation between the applied pressure and the $V_{out}$ measurement is first validated by pressing a single mechanism above the sensor. The evaluation in Figure 24 shows the straight-line relationship between the $V_{out}$ (expressed by a 16-bit number) and the weight load in grams distributed over an area of 10 $mm^2$. Different pinpoint areas were tested, nevertheless, tip areas lower than 8-10 $mm^2$ had small shifts while the load was applied, changing the ADC readout. The tip area covers 2 adjacent sensing pixels, but this provides consistent data. For the evaluation, $V_{test}$ (eq. 6) is set to 0.4V. Higher $V_{test}$ raises the $V_{out}$ to the maximum ADC value when the minimum possible pressure is applied to the sensor pixel while small $V_{test}$ values (in the range of 10 mV) make ghost activations due noise susceptibility. It is quite important to properly calibrate $V_{test}$ otherwise data quality for training the ML approach is low or cross-talk information might be limited.

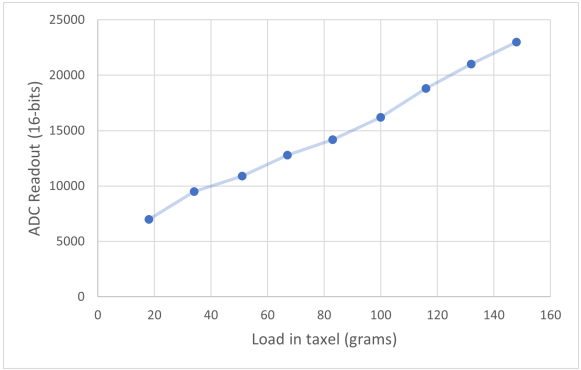



Fig. 24: $V_{out}$ response with $N = 2$. $V_{test} = 400mV$

In addition to accurate calibration of $V_{test}$, it is crucial to correctly size the resistance $R_k$. Otherwise, an incorrect sizing of $R_k$ can cause the $V_{out}$ readout to reach the maximum 16-bit value even when small loads are applied, leading to inadequate

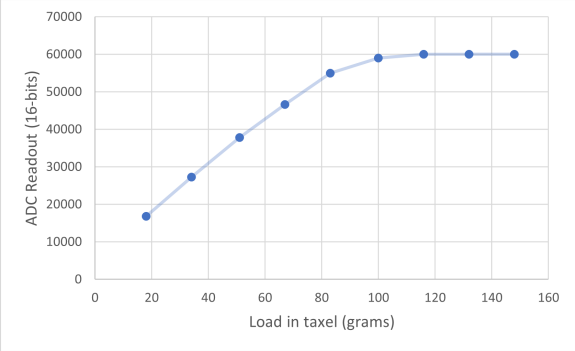



Fig. 25: $V_{out}$ response with $R_k$ incorrectly sized or high $V_{test}$

sensing of high pressures. Figure 25 shows how ADC gets to the maximum value when the load in the sensing pixel is more than 100 grams. In our acquisition circuit $R_k$ value is set at 2 $k\Omega$ since the range of human pressure when a sensing pixel is pressed goes from 3 $k\Omega$ to 500 $\Omega$. The $R_k$ sizing is validated through an LTSpice simulation (Figure 26) which considers an estimated value for $R_i$ (cross-talk equivalent resistance) as well.

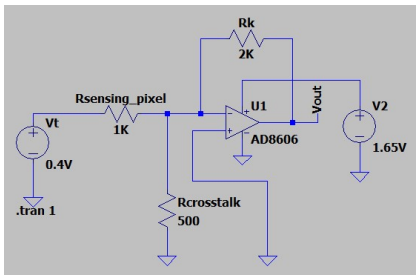



Fig. 26: LTSpice simulation: Simplified Model

Since the alternative current paths only significantly happen when more sensing pixels are activated in the row path. The evaluation consists of more and more sensing pixels pressed over a row with the same pressure. In order to validate equation 6, ten mechanisms were placed in the same row, one next to another (Figure 27a). The same pressure is applied in mechanisms, and data is subsequently recorded. The experiment is repeated twice taking ten measurements per experiment. Then, one mechanism is removed and the process is repeated until just one pressure mechanism is left (Figure 27b). The process is again repeated over 7 different areas in the sensor to reduce the misalignment error. This gives a total of 140 performed experiments and 1400 data points to determine behavioural curves.

Once the 1400 data points are collected, these are plotted according to the number of sensing pixels ($N$) that are covered in PRS array. Figure 28 shows how ADC 16-bit value slightly drops ($V_{out}$) when $N$ is increased. This sensor behaviour in function of $N$ is consistent with the expected behaviour predicted by the interactive model. Lastly, the interactive model parameter scenario is set as shown in Table V with logical values to check if it matches the experimentally obtained data

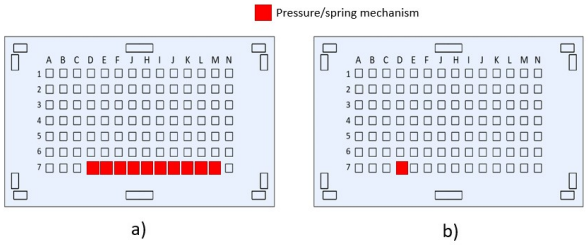



Fig. 27: Experiment (a) initial setup and (b) final setup for one area in the sensor

(Figure 29). The observed data aligns with the predicted by the model, indicating that the model accurately captures the behaviour of the sensor under cross-talk conditions.

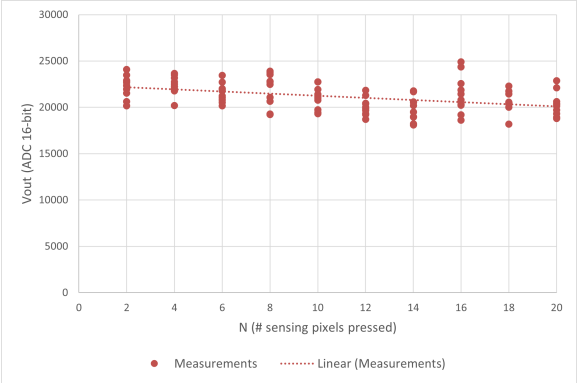



Fig. 28: $V_{out}$ drops due N increase. Constant load of 132 gr.

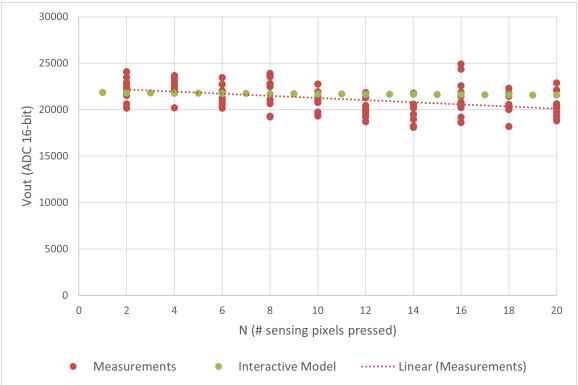



Fig. 29: Interactive model matching sensor's data

| | |
|---|---|
| $A$ | 100,000 |
| $R_k$ | 2,000 |
| $R_i$ | 1,000 |
| $R_{taxel}$ | 1,000 |
| $V_{test}$ | 0.4 |

TABLE V: Interactive model parameters

*1) Circuit protection $R_v$ effect at test voltage $V_{test}$ output:* At the output of $V_{test}$, a resistor $R_v$ was previously incorporated to provide circuit protection. The equivalent resulting circuit is shown in Figure 30. This makes the cross-talk effect caused by $R_m$ (shown in Figure 21) more considerable. The relation between $R_m$ and $R_v$ causes a voltage divider at the input of the sensing pixel resistance $R_s$, resulting in a considerable drop in $V_{in}$ if $R_v$ is much higher than $R_m$:

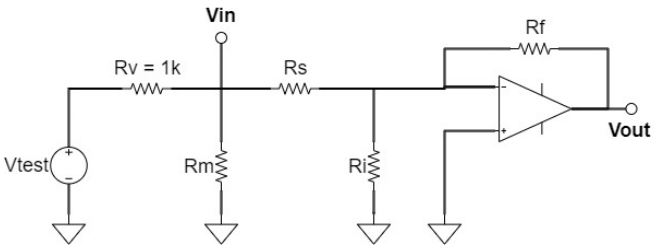



Fig. 30: Model considering $R_m$ and $R_v$

$$V_{\text{in}} = V_{\text{test}} \frac{R_m}{(R_m + R_{\text{v}})} \tag{7}$$

$$V_{\text{in}} = V_{\text{test}} \frac{1}{(1 + 1000)} = V_{\text{test}}(0.0009)$$

In the performed experiments, $R_v$ significantly amplifies the cross-talk effect, leading to a sharper decline $V_{out}$ as $N$ increases in the experimental evaluation. The impact of $R_v$ can be observed in Figure 31, where $V_{out}$ drops as $N$ increases. When $R_v$ is implemented in the circuit, the role of $R_m$ becomes more significant. The experiments also revealed that the influence of $R_m$ is greater than initially anticipated if $R_v$ is implemented. Additionally, when the input voltage $V_{in}$ is low, it becomes more susceptible to steeper declines in $V_{out}$ when more sensing pixels are pressed in the same row. However, in the new design iteration, the steeper decline caused by cross-talk is mitigated by removing $R_v$.

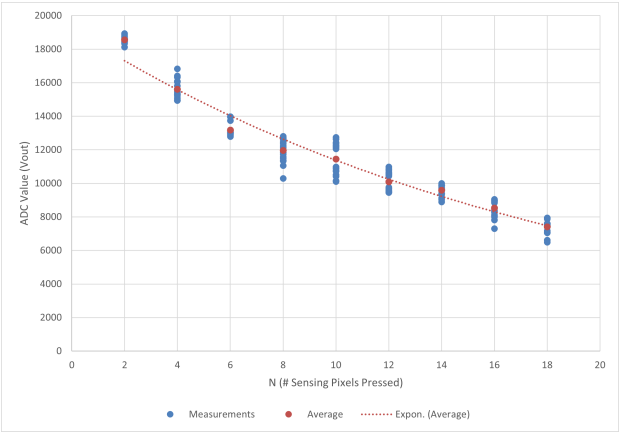



Fig. 31: Alternative type of cross-talk with $R_{test}$ incorrect sizing

## VI. Cross-talk Computational Compensation Method

We implement a machine learning method using artificial neural networks (ANN) to compensate for the cross-talk effect. A well-trained Machine Learning model can effectively learn the complex relationships between sensor measurements and generate compensation values. One advantage of using neural networks is that they can be trained on large datasets, which enables them to capture the intricate relationships between sensor measurements and the expected pressure value even in the presence of cross-talk. The method approach involves collecting data from the sensor array under various conditions. A known labelled reference pinpoint pressure is utilised, and this reference is deliberately disturbed by cross-talk environments, such as applying excessive pressure to a different section in the same row as the reference point. Then, the data is used to train a supervised machine learning model that can predict the estimated load (or pressure) on the sensing pixel given the $V_{out}$ measurements (Figure 32).

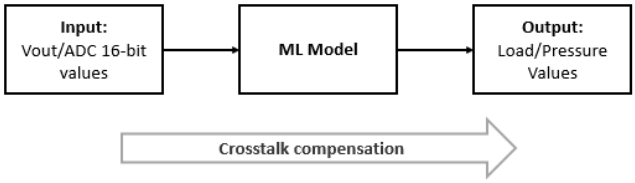



Fig. 32: Overview cross-talk compensation

The data have to be high-quality data that accurately describes the sensor's behaviour. This requires control over the experimental conditions and the use of appropriate data acquisition and processing techniques to minimise outliers, noise, and other possible sources of error. Figure 33 shows the process used to train the compensation method.

In order to implement the compensation method, it is essential to consider the next conditions:

- The 16-bit ADC values (0 to 65,535) are available for all the rows and columns.
- Cross-talk information has to be present in ADC $V_{out}$ sensor's array measurements.
- Outliers should be filtered before inputting the data into the model.
- The sensor must be capable of producing high-dimensional data that can be used to train a machine learning mode, and a sufficient amount of data should be collected under a wider range of conditions.

The compensation method calculates the value-sensing pixel pressure value considering all the values in the same row which are linked to the parasitic cross-talk resistances. The method is applied sequentially to every element in the sensor array. Two important aspects to consider are the data filtering and the data management to input into to ML model:

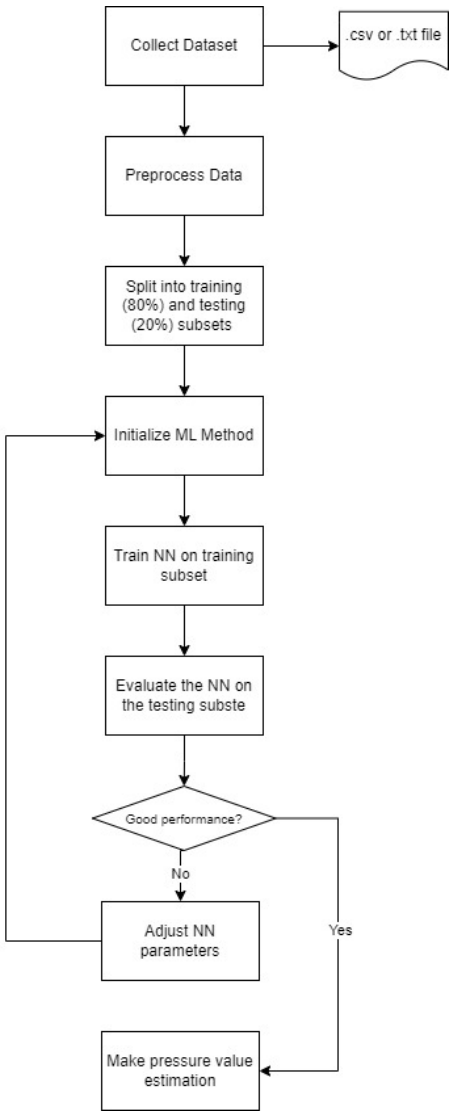



Fig. 33: Flow chart for training ML models

*1) Data filtering:* As seen in Table IV some outliers are present in the measurements, usually these values either go to the maximum or to the minimum of the 16-bit number of the ADC. This makes it relatively simple to filter the data by placing a mask over it. The mask filters the values that are close to low and high extremes. After removing outliers, the data set is built to train ML models.

*2) Data management:* The large dataset used to train the model is stored in a *.csv* file. The dataset is constructed carefully, separating data that is the input and its corresponding labels. As seen in Table VI, the first column in the data set contains the sensing pixel ADC value, the consequent columns until the second-to-last column are the values in the same row, and the last one contains the ground truth load. For this application, the order of row values does not matter for how you input information (second column in Table VI), therefore a random shuffling can be done to augment the dataset for training and testing.

| Taxel value | Taxel Value Pos [0-128] | Load (grams) |
|---|---|---|
| 12520 | 22520... 48201 | 18 |
| 25840 | 23840... 14436 | 67 |
| 10840 | 19561... 65820 | 18 |
| 39840 | 5906... 3589 | 165 |
| 254 | 252... 48201 | 0 |

TABLE VI

### *A. Neural Network Architecture*

The supervised machine learning model used is a basic Multi-Layer Perceptron (MLP) Feed-forward ANN. MLPs are effective at predicting values in a regression problem due to their ability to capture complex linear and nonlinear relationships between input variables and output variables. With the flexibility to design MLPs, they can be adaptable if the dataset for training is increased and can scale well on large datasets. Figure 34 illustrates the architecture used for the NN net.

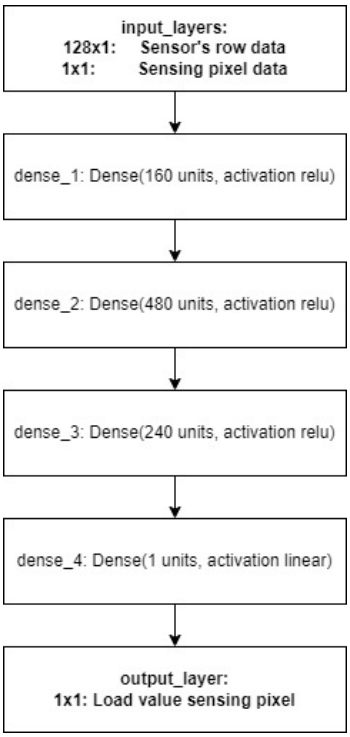


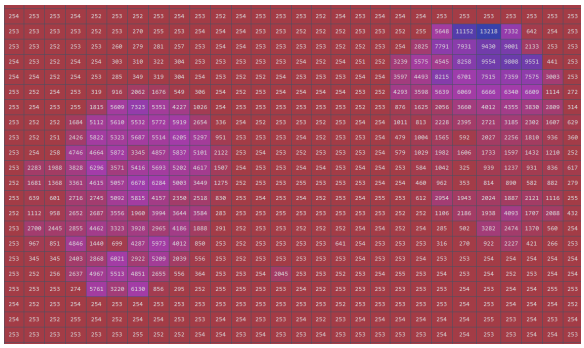

Fig. 34: Neural network net architecture

After defining the NN architecture, an algorithm is developed to perform the compensation. The algorithm reads the sensor's measurements and structures the data in such a way that can be input to the NN model. Algorithm 1 illustrates the compensation method programmed in Python. It uses the "*scikit-learn*" library and its class "*MLPRegressor*" that implements the multi-layer perceptron model that is commonly used for regression tasks. Sklearn "*MLPRegresor*" can provide a high degree of accuracy, handle non-linear datasets, and be easily set in Python.

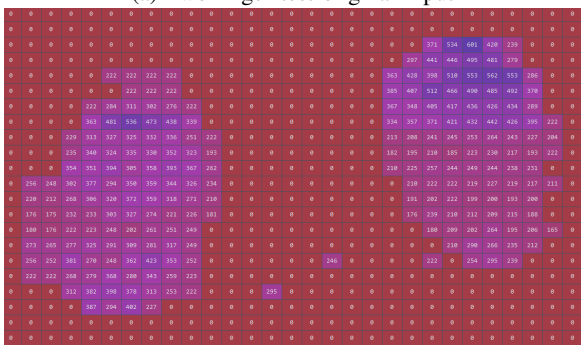
**Input:** 64x128 Matrix ADC values readout
**Output:** 64x128 Matrix load estimated values

```
Import MLP-trained model;
Read PRS measurement 64x128 dataset;
for every row do
    Make ML input matrix structure;
    if ADCValue ≤ 300 then
        Count as pressed pixel;
        for every pixel do
            Input previous matrix into the model;
            Store the pressure output of the model;
        end
    end
    else
        for every pixel do
            Fill zero load in that pixel;
        end
    end
    Reconstruct row with new values;
end
Reconstruct 64x128 matrix with new rows;
Interpolate image for displaying;
Print image result;
```

**Algorithm 1:** Pixel-by-pixel compensation pseudocode



(a) Two finger test original input



(b) Two finger test after compensation

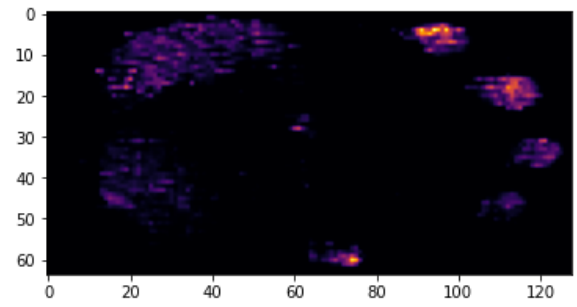
Fig. 35: Compensation and pressure estimation process

### B. Compensation Results

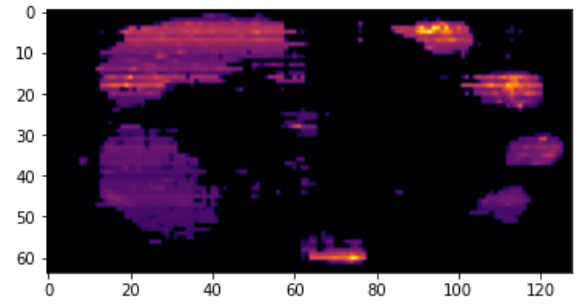
After the compensation has been executed, the image is reconstructed with the estimated load values per sensing pixel. Figure 35 illustrates how readout pixel values change after the compensation process. The first matrix (Figure 35a) shows the ADC values taken by the sensor readout, and the output matrix (Figure 35b) shows how these values change to estimated load values (in grams). These values can be transformed to pressure values (in MegaPascals) by just dividing all matrix values by 10 $mm^2$. It is evident that the finger on the right side is affected by cross-talk, particularly in the rows where both fingers coincide. This results in a drop in the measurement of the lower part of the finger. However, after the compensation process, the values become more uniformly distributed, indicating some level of cross-talk compensation.

Figure 36 shows the compensation result for a hand readout. It is seen how sensing pixels that coincide with highly pressured points are higher compensated. High-pressured points create higher cross-talk affectations, therefore pixels on that row need to be better compensated. During the test, the left side of the palm pressed harder on the sensor, this is well seen in the image top side output Figure 36b.

In order to evaluate the compensation, the load value of the test data (ground truth) is compared to the output of that data input to the neural network model. The difference average variation between ground truth and the estimation is $\pm 10$ where the possible range can go from 0 to 600 grams.




(a) Input matrix: Readout from sensor




(b) Output matrix: Compensated imaging

Fig. 36: Crosstalk compensation for a hand palm readout

## VII. Discussion

Accurate measurement acquisition for the analysis was a significant challenge. In order to rely on the measurement, it was necessary to average multiple experiments. This approach was essential to establish confidence in the measurement results. To reduce the time required for multiple experiments, it is recommended to design a high-precision system for applying the loads to the sensor. The sensor initially exhibited reliability issues primarily due to connectivity problems, resulting in the need for troubleshooting. Resolving the problem of highly noisy measurements required a significant amount of effort and work. It is suggested to use a different connector between the flexible PCBs and to design a structure that prevents bending on the region where rigid components are soldered. This future approach will help to mitigate the occurrence of misconnections caused by imperceptibly broken soldered paths.

In terms of the circuit configuration, an unrecognised source of cross-talk was identified, arising from the presence of a resistance connected to the output of the voltage source $V_{test}$. This introduced a distinct cross-talk type that differed from what was described in the extended mathematical model. The coexistence of these cross-talk sources can potentially mislead interpretations of the results when compared to the mathematical model. However, by eliminating the resistor, the cross-talk induced by this resistor was successfully suppressed, significantly reducing the observed cross-talk. Great care must be taken when sizing the values for $R_k$ and $V_{test}$. Poorly dimensioned values can make it impossible to read the pressures in the human range, even if the pattern is visually shown in the imaging. To ensure accurate sizing, it is crucial to consider the observed resistance value $R_{taxel}$ when a finger is pressed against it for a single sensing pixel. It is also suggested to explore other Velostat materials in order to have a larger range of sensible pressures, meaning to have a thicker piezoresistive with a larger linear behaviour range.

The table setup serves the purpose of data collection to study the behaviour of the sensor and train the machine learning algorithm. It allows different configurations to provide more accurate load placement with the spring mechanism, consequently minimising misalignment and drift issues during load application. To mitigate human error during spring compression, an automated system can be employed to ensure consistent and accurate load application. The compensation method has certain limitations that should be considered. One of the main limitations is the processing time required, as the method needs to be executed for every sensing pixel in the PRS array, which can affect real-time cross-talk compensation. Another limitation arises from the readout data, as values larger than 65,535 ADC value cannot be captured, making the cross-talk level unknown beyond this threshold. To address this, it is suggested to maintain a low sensitivity level by setting a low $V_{test}$ voltage. This ensures that most of the cross-talk information is captured in the data, even when the sensor is subjected to extremely high pressure.

## VIII. Conclusion

The proposed approach for cross-talk compensation has several advantages over traditional methods for compensating for the cross-talk effect. First, do not need to increase the complexity of the circuit, so this allows us to keep a portable version of the e-Cone and keep the sensor cost low. Second, it is highly flexible and adaptable, allowing for easy updates and modifications as new data becomes available. Due to some inconsistencies in data, like outliers in sensor readings, caution should be exercised when interpreting the results obtained from the sensor. Therefore, we recommend that future cross-talk studies employ a more reliable sensor or conduct additional tests to ensure result accuracy. Then, collect more data to train a model. In addition to that, new machine-learning methods can be explored to execute the compensation in real-time.

The choice of the MLP algorithm proved to be a strong choice for our pressure estimation problem. Its ability to capture complex non-linear relationships, learn relevant features from the data, and generalize to unseen instances resulted in valid estimations and showcased the effectiviness of MLPs in adddresing this specific problem. Overall, the proposed approach represents an advance in the field of piezoresistive sensor arrays and has the potential to significantly improve the accuracy and reliability of sensor measurements in a wide range of applications, including the e-Cone sensor.


## Acknowledgements

I am deeply grateful to my loving family for their unwavering support, my exceptional supervisors for their invaluable criticism and expert guidance, and my girlfriend and friends for their assistance in conducting the necessary experiments that made this project possible. Their belief in me, patience, and constant encouragement have been integral to the completion of this thesis, and I am truly blessed to have such a strong support system from the University of Twente by my side. Finally, I would like to express my gratitude to the academic community for providing the resources, facilities, and intellectual environment necessary for the successful completion of this work.


## References


[1] A. Fatema, I. Kuriakose, D. Devendra, and A. M. Hussain, "Investigation of the mechanical reliability of a velostat-based flexible pressure sensor," in *2022 IEEE International Conference on Flexible and Printable Sensors and Systems (FLEPS)*, 2022, pp. 1–4.

[2] S. Suprapto, A. Setiawan, H. Zakaria, W. Adiprawita, and B. Supartono, "Low-cost pressure sensor matrix using velostat," in *2017 5th International Conference on Instrumentation, Communications, Information Technology, and Biomedical Engineering (ICICI-BME)*, 2017, pp. 137–140.

[3] A. Dzedzickis, E. Sutinys, V. Bucinskas, U. Samukaite-Bubniene, B. Jakstys, A. Ramanavicius, and I. Morkvenaite-Vilkonciene, "Polyethylene-carbon composite (velostat®) based tactile sensor," *Polymers*, vol. 12, no. 12, p. 2905, 2020.

[4] S. Müller, D. Seichter, and H.-M. Gross, "Cross-talk compensation in low-cost resistive pressure matrix sensors," in *2019 IEEE International Conference on Mechatronics (ICM)*, vol. 1, 2019, pp. 232–237.

[5] L. Harris and M. Oliver, "Design and fabrication of a piezoresistive tactile sensor for ergonomic analyses," Ph.D. dissertation, 01 2014.

[6] B. B. Heather markham, in *Development of a skin for intuitive interaction with an assistive robot*, 2009. [Online]. Available: https://doi.org/10.1109/IEMBS.2009.5334522

[7] J.-F. Wu, "Scanning approaches of 2-d resistive sensor arrays: A review," *IEEE Sensors Journal*, vol. 17, no. 4, pp. 914–925, 2017.

[8] A. S. Warnakulasuriya, N. Y. Dinushka, A. A. C. Dias, H. P. A. R. Ariyarathna, C. Ramraj, S. Jayasinghe, and A. C. De Silva, "A readout circuit based on zero potential crosstalk suppression for a large piezoresistive sensor array: Case study based on a resistor model," *IEEE Sensors Journal*, vol. 21, no. 15, pp. 16 770–16 779, 2021.

[9] L. Yuan, H. Qu, and J. Li, "Velostat sensor array for object recognition," *IEEE Sensors Journal*, vol. 22, no. 2, pp. 1692–1704, 2022.

[10] R. Dodds, H. Syddall, R. Cooper, M. Benzeval, I. Deary, E. Dennison, G. Der, C. Gale, H. Inskip, C. Jagger, T. Kirkwood, D. Lawlor, S. Robinson, J. Starr, A. Steptoe, K. Tilling, D. Kuh, C. Cooper, and A. Aihie Sayer, "Grip strength across the life course: Normative data from twelve british studies," *PloS one*, vol. 9, p. e113637, 12 2014.

[11] S. Solgaard, B. Kristiansen, and J. S. Jensen, "Evaluation of instruments for measuring grip strength," *Acta Orthopaedica Scandinavica*, vol. 55, no. 5, pp. 569–572, 1984, pMID: 6507083. [Online]. Available: https://doi.org/10.3109/17453678408992963

[12] Z. Hu, D. Rajendran, F. Wendler, R. Ramalingame, and O. Kanoun, "Evaluation of the cross talking effect in piezoresistive tactile sensor matrices," in *2018 15th International Multi-Conference on Systems, Signals Devices (SSD)*, 2018, pp. 632–635.

[13] H. W. Luuk Spreeuwers, in *A high resolution pressure sensor for measurement of grip force*, 2019.

[14] Y. Aldien, D. Welcome, S. Rakheja, R. Dong, and P.-E. Boileau, "Contact pressure distribution at hand–handle interface: role of hand forces and handle size," *International Journal of Industrial Ergonomics*, vol. 35, no. 3, pp. 267–286, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169814104001623

[15] K. Smit, in *A high resolution grip strength measuring system for rehabilitation of hand conditions*, 2021.

[16] M. Hopkins, R. Vaidyanathan, and A. H. Mcgregor, "Examination of the performance characteristics of velostat as an in-socket pressure sensor," *IEEE Sensors Journal*, vol. 20, no. 13, pp. 6992–7000, 2020.

[17] T. D'Alessio, "Measurement errors in the scanning of piezoresistive sensors arrays," *Sensors and Actuators A: Physical*, vol. 72, no. 1, pp. 71–76, 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924424798002040

## IX. Appendix

### A. Sensing pixel analyser

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Create a 128x64 matrix of zeros
matrix = np.zeros((64, 128))

# Read data directly from a text file
data = np.genfromtxt('FORGRAPHS/C7_400mV_NEW.
    txt',
                     delimiter=',',
                     autostrip=True)
data = data[:,:-1]

# Select sensing pixel
# IMPORTANT: IN SENSOR'S ACQUISITION
    PROGRAMMING ROW AND COLUMNS ARE SWAPPED
# THERE ARE 128 ROWS, 64 COLUMNS
row_index = 58
col_index = 2
pressure = 0.5 #MPa

# Access the value at the specified row and
    column
value = data[row_index, col_index]
print(value)

# Define step size
step_size = 64 #TO KEEP IT FIXED, 65 for CSV
    files, 64 for TXT files

# Extract the column index
column = data[:, col_index]

# Extract every 64th or 65th element from the
    column
new_column = column[row_index::step_size]

# Export new_column to a Excel File
pixelvalue = pd.DataFrame(new_column)
pixelvalue.to_excel(r'D5_p.xlsx', sheet_name='
    1', index=False)

# Extract the whole row every 64 rows
alldatacolumn = data[row_index::step_size]

# Create an array of x values from 0 to n
x = np.arange(len(new_column))

# Plot the values
fig1 = plt.figure(1)
plt.plot(x, new_column)

# Set the axis labels and title
plt.xlabel('Iteration')
plt.ylabel('ADC Value')
plt.title(f'ADC Values from Taxel in Row {
    col_index:.2f} and Column {row_index:.2f}
    at {pressure:.2f} MPa')

# Set the threshold value
threshold = 300

# Create a boolean array for values above the
    threshold
mask = new_column < threshold

# Use the boolean array to remove values from
    the column
new_column_filtered = new_column[~mask]
new_column_filtered = new_column_filtered.T

# Create an array of x values from 0 to n
x2 = np.arange(len(new_column_filtered))

# Plot the values
fig2 = plt.figure(2)
plt.plot(x2, new_column_filtered)

# Set the axis labels and plot
plt.xlabel('Iteration')
plt.ylabel('Value')
plt.title(f'ADC Values Filtered from Taxel in
    Row {col_index:.2f} and Column {row_index
    :.2f} at {pressure:.2f} MPa' )

# Display the plot
plt.show()

# Percentage data filtered
percetange_filtered = 100 - ((len(
    new_column_filtered) / len(new_column))
    *100)
print('percetange filtered:' ,
    percetange_filtered)
```

### B. ML model design and training

```
import numpy as np
import pandas as pd
from sklearn.neural_network import
    MLPRegressor
from sklearn.model_selection import
    train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import
    mean_absolute_percentage_error
from sklearn.preprocessing import
    StandardScaler
import pickle

# From a CSV file
data = pd.read_csv("DataMLPTraining.csv")
data = np.matrix(data)
data = np.asarray(data, dtype=np.float32)
prev_matrix = data

# Function for shuffling each row of the
    matrix, except the first and last element
def shuffle_row(row):
    mid = row[1:-1]
    np.random.shuffle(mid)
    return np.concatenate((row[:1], mid, row
        [-1:]))

# Create 100 new matrices with shuffled rows
new_matrices = []
for i in range(100): # define the number of
    shuffling
```

```
    shuffled_matrix = np.apply_along_axis(
        shuffle_row, 1, prev_matrix)
    new_matrices.append(shuffled_matrix)

# Stack all matrices vertically
stacked_matrix = np.vstack([prev_matrix] +
    new_matrices)

x_pre = stacked_matrix[:, :129]
y = stacked_matrix[:, -1]

scaler = StandardScaler()
x = scaler.fit_transform(x_pre)

# Split the dataset into training and test
    sets
x_train, x_test, y_train, y_test =
    train_test_split(x, y, test_size=0.1,
    random_state=1)

# Initialize an MLP model with hidden layers
model = MLPRegressor(hidden_layer_sizes
    =(160,480,240), activation='relu', solver=
    'adam', max_iter=2000)

# Train the model with the sample data
model.fit(x_train, y_train)

# Save the model to a file
filename = 'MLPModel129to1_2.sav'
pickle.dump(model, open(filename, 'wb'))

# Evaluate the MLP model
y_pred = model.predict(x_test)
mse = mean_squared_error(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test,
    y_pred)
print("Mean Squared Error:", mse)
print("Mape:", mape)

# Calculate errors
errors = list()
for i in range(len(y_test)):
 # calculate error
 err = abs((y_test[i] - y_pred[i]))
 # store error
 errors.append(err)
 # report error
 #print('>%.1f, %.1f = %.3f' % (y_test[i],
     y_pred[i], err))

# Average errors
average = sum(errors)/len(errors)
print("Average:", average)
```

### C. Computational compensation method

```
import pickle
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import
    StandardScaler
from PIL import Image

scaler = StandardScaler()

def interpolate_heatmap(heatmap_array, factor)
    :
    # Convert the array to a Pillow Image
    heatmap_image = Image.fromarray(
        heatmap_array)

    # Calculate the new dimensions
    width, height = heatmap_image.size
    new_width = width * factor
    new_height = height * factor

    # Resize the image using bilinear
        interpolation: BILINEAR, BICUBIC,
        NEAREST
    interpolated_image = heatmap_image.resize
        ((new_width, new_height), Image.
        BILINEAR)

    # Convert back to NumPy array
    interpolated_heatmap = np.array(
        interpolated_image)

    return interpolated_heatmap


# Import ML trained model
filename = 'MLPModel129to1_2.sav'
model = pickle.load(open(filename, 'rb'))

# Read Sensor measurement 64x128
data = np.genfromtxt('HANDREADOUT/CARLOSHAND4.
    txt', delimiter=',', autostrip=True)
data = data[:,:-1]
data = np.matrix(data)
data = np.asarray(data, dtype=np.float32)

# Take just 1 sensor mattress readout
data1 = data[:64, :]
data1 = np.round(data1)

# Creat mask with pressed pixels
mask = np.where(data1 > 260, 1, 0)

# generate the heatmap
heatmap = plt.imshow(data1, cmap='inferno')
# plt.colorbar(heatmap) # add a colorbar to
    the heatmap
# show the plot
plt.show()

# Save features in a matrix
x_testin = np.zeros((0, 129))

# For every pixel on that row...
for i in range(data1.shape[0]):  # iterate
    over rows
    for j in range(data1.shape[1]):  # iterate
         over columns
        if data1[i,j] > 350: # filter to
            compensate only pressed pixels
            cell_value = data1[i, j]
#            print(cell_value) # For Debugging
            x_test = np.array([cell_value])
            x_test = np.concatenate((x_test,
                data1[i,:]), axis=0)

            x_testin = np.vstack((x_testin,
```

```
                x_test))
        else: # for pixels not pressed, set
            pressure to 0
            x_test = np.zeros((1, 129))
            x_testin = np.vstack((x_testin,
                x_test))
        #print(N, data1[i,j], Pt)
        #print(y_pred) # For Debugging
        #print(x_test) # For Debugging

# Calculate output & Store the output of the
    model
x_testin_nonorm = x_testin # TO DEBUG
x_testin = scaler.fit_transform(x_testin)

y_pred = model.predict(x_testin)
y_pred = np.round(y_pred, decimals=0)

# Reconstruct 64x128 matrix
matrix_out = y_pred.reshape((64, 128))

# replace all occurrences of 163 with 0
matrix_out = matrix_out*mask

# Set all negative values to zero using clip
    function
matrix_out = np.clip(matrix_out, 0, None)

# generate the heatmap
heatmap2 = plt.imshow(matrix_out, cmap='
    inferno')
# add a colorbar to the heatmap
#plt.colorbar(heatmap2)
# show the plot
plt.show()

# Image interpolation for Display
heatmap_array = heatmap2.get_array()
interpolation_factor = 10
interpolated_heatmap = interpolate_heatmap(
    heatmap_array, interpolation_factor)
# Display the interpolated heatmap
plt.imshow(interpolated_heatmap, cmap='inferno
    ')
#plt.colorbar()
plt.show()
```