

# Optimizing BioTac Simulation for Realistic Tactile Perception

Wadhah Zai El Amri

L3S Research Center, Leibniz Universität Hannover  
Hannover, Germany  
0000-0002-0238-4437

Nicolás Navarro-Guerrero

L3S Research Center, Leibniz Universität Hannover  
Hannover, Germany  
0000-0003-1164-5579

**Abstract**—Tactile sensing presents a promising opportunity for enhancing the interaction capabilities of today’s robots. BioTac is a commonly used tactile sensor that enables robots to perceive and respond to physical tactile stimuli. However, the sensor’s non-linearity poses challenges in simulating its behavior. In this paper, we first investigate a BioTac simulation that uses temperature, force, and contact point positions to predict the sensor outputs. We show that training with BioTac temperature readings does not yield accurate sensor output predictions during deployment. Consequently, we tested three alternative models, i.e., an XGBoost regressor, a neural network, and a transformer encoder. We train these models without temperature readings and provide a detailed investigation of the window size of the input vectors. We demonstrate that we achieve statistically significant improvements over the baseline network. Furthermore, our results reveal that the XGBoost regressor and transformer outperform traditional feed-forward neural networks in this task. We make all our code and results available online on [https://github.com/wzaielamri/Optimizing\\_BioTac\\_Simulation](https://github.com/wzaielamri/Optimizing_BioTac_Simulation).

**Index Terms**—BioTac, XGBoost, Transformer, Tactile Perception

## I. INTRODUCTION

Tactile sensing sensors offer robots valuable information that can be used to enhance and complement knowledge coming from other modalities such as vision or audio, especially in situations where this knowledge is entirely or partially not available [1], [2].

In situations where the sensor is unavailable or experiment repetitions are costly, the value of a reliable, real-time simulation becomes evident. Such a simulation can effectively estimate sensor outputs for various touch scenarios. Such simulation would offer a good alternative to gathering data in different setups and environments [1]. However, simulations only approximate real-world data, and models trained on this data alone can typically not be directly used in real robots. This problem is also known as the reality gap, and it can be mitigated with techniques such as sim2real [3] or continual learning [4]. Nevertheless, the quality of the simulated data should be as high as possible to minimize the reality gap and increase the success of other modules down the pipeline.

Various studies have presented simulations for a range of tactile sensors, including TACTO [5] for vision-based sensors like DIGIT [6] and OmniTact [7], touch simulation for fabric-based tactile sensors [8], a simulation for the iCub skin [9], and several simulations for the BioTac sensor [10]–[12].

However, in this paper, we focus on the BioTac [13], [14], one of the most widely used tactile sensors [15]. It consists of a rigid core covered by an elastomeric skin filled with an incompressible conductive fluid. This sensor proved to be helpful in different tasks, such as grasp stability [16], object identification [17], localizing artificial tumors [18], [19], etc.

Some simulations of the BioTac sensor exist. Ruppel et al. [10] collected a real-world dataset with the BioTac 2P sensor [20] mounted on a Shadow Robot Hand [21]. The dataset captures the tactile sensor readings while touching with different forces, positions, and orientations, an indenter with a spherical tip of radius equal to 2 mm, attached to a calibrated force-torque sensor (ATi nano17e [22]). They propose a neural network model that can estimate electrodes, vibration, and pressure signals using the force and position vector of the contact point alongside temperature values. However, their approach requires using the temperature values in the input vector, which are not available in rigid body simulation environments like Gazebo [23].

Narang et al. [11] developed a finite element model (FEM) that can mimic the deformation behavior of the elastomeric skin and the liquid gel inside. From the point cloud of the FEM nodal field data, a neural network based on the PointNet++ architecture [24] is used to estimate the electrode values of the BioTac 2P [20]. To validate the FEM model and train the neural network, they collected data on different trajectories using nine different indenters interacting with three different tactile sensors. The FE simulation takes 7 minutes to simulate one single trajectory, which makes it a big drawback. Later, they improved the approach using the NVIDIA Isaac Gym simulator [25], achieving a 75 times improvement [26]. However, a single trajectory simulation of 6 mm takes 5.57 s to simulate.

Furthermore, the authors improved the electrodes’ estimation, by using a self-supervised latent learning network that maps between the FEM Mesh and electrode signals. Despite these improvements, this solution is still not fast enough to be used in real-time systems. Moreover, this solution estimates only the electrodes’ values and not the pressure or vibration signals recorded by the BioTac sensor, which are valuable modalities when used in grasping and manipulation tasks [27].

Zaoata-Impata and Gil [12] used vision to estimate the electrode values of the BioTac SP sensor [28]. A modified Semi-Regression Generative Adversarial Network (SR-GAN) [29]

arXiv:2404.10425v1 [cs.RO] 16 Apr 2024

was used to synthesize electrode values out of point cloud representation of grasped deformable daily objects and their grasping data by using labeled and unlabeled input data. The authors collected 4000 samples of grasping positions, electrodes, and pressure values of two BioTac SP sensors mounted on a Shadow Robot Hand [21] and 3D point clouds recorded with an Intel RealSense D415 depth camera. Since collecting data is not trivial, a generator was used to generate synthetic BioTac data for the point clouds and the grasping positions in order to augment the dataset and improve the learning of the discriminator, whose task was to estimate the correct BioTac outputs. The unlabeled samples boosted the performance of the network. In addition, this work sheds light on the possibility of using vision, in this case through point clouds, to estimate signal data for the BioTac sensor. However, this implementation only used 4000 samples.

Since vibration and pressure readings add valuable information when used for manipulation tasks [27], we must simulate these alongside electrode values. The solution of Narang et al. [26] only estimates the electrode signals and is comparatively slow. In comparison, Zaoata-Impata and Gil [12] implementation uses a small dataset. Considering these factors, we opt for the solution provided by Ruppel et al. [10].

In the next Section II, we revisit the research conducted by Ruppel et al. [10]. Within that section, we first delve into the details of their work. Subsequently, in Section III, we thoroughly investigate two directions to improve the simulation further.

## II. BIOTAC SENSOR SIMULATION

The simulation provided by Ruppel et al. [10] is based on a neural network trained with a real-world dataset to predict the BioTac sensor signals. This dataset includes data from approximately one hour of recording sampled at 100 Hz. It consists of the complete BioTac output values, i.e., 19 electrode voltage  $e_1, \dots, e_{19}$ , absolute and dynamic fluid pressure  $pdc$  and  $pac$ , temperature  $tdc$  and heat flow  $tac$ . The dataset also includes the BioTac's and indenter's position and orientation recorded with a vision tracking system [30]. For the data pre-processing pipeline, the position of the indenter's tip is calculated using transformation matrices in the BioTac coordinate frame, represented in Figure 1.

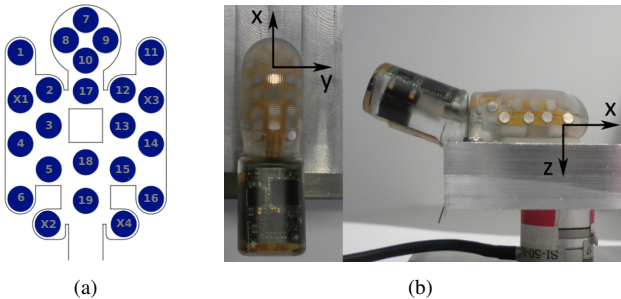


Fig. 1. (a): Map of the electrodes on the BioTac sensor [31]. (b): BioTac sensor and the position/ orientation of the coordinate frame [31].

These transformation matrices are calibrated to adjust for systematic offsets by reducing the error between the probes' positions and the BioTac surface. In other words, probes with light touch contact at the end of a contact cycle and with force measurements below 0.3;  $N$  are gathered, and their distance to the skin surface of the BioTac is calculated with the transformation matrices. In such cases, that distance should be equal to 0. Due to calibration issues, this distance deviates from 0, 5.039 mm on average. While iterating for 1000 steps and adjusting these transformation matrices, i.e., adding and subtracting small values, the distance error is reduced to 0.444 mm on average for all these selected probes.

Two deep neural networks,  $A$  and  $B$ , were proposed to simulate the sensor outputs at timestep  $t = T$ . Both networks take as input the  $x$ ,  $y$ , and  $z$  contact position of the indenter's tip in the BioTac coordinate frame at timestep  $t = T$ ,  $F_x$ ,  $F_y$ , and  $F_z$  values of the force sensor at timesteps  $T$ ,  $T - 10$ , and  $T + 10$  and the temperature directly taken from BioTac readings at timestep  $t = T$ , i.e., an input vector of length 13.

Network  $A$  is a dense neural network with five hidden layers and over 6 million parameters, employing a pre-mapping fusion strategy [1]. On the other hand, Network  $B$  is a more compact neural network with around 800 thousand parameters and adopting a midst-mapping fusion strategy [1]. Network  $B$  consists of three separate dense layer columns that merge into a dense layer. Positional, force, and temperature values undergo separate processing within these columns.

Both networks were trained with the loss function presented in Equation 1, which consists of the summation of the mean absolute error (MAE) and the mean squared error (MSE).

$$Loss = \frac{\sum_{i=1}^n (|y_i - \hat{y}_i| + (y_i - \hat{y}_i)^2)}{n} \quad (1)$$

The authors report achieving 9.3% normalized mean absolute error (MAE) over all channels calculated on the  $z$ -score normalized outputs of the neural network, i.e., not in the original scale between 0 and 4095, with Network  $B$ . However, Network  $A$  reached 11.3% MAE. In both cases, the reported results values consider only one of the two dynamic pressure channels ( $pac_0$ ) because both channels are nearly identical. The reported results also omitted the heat flow channel ( $tac$ ). During the neural network training phase, both channels, i.e.,  $pac_1$  and  $tac$ , were utilized without justification for this approach, resulting in an output vector comprising 23 channels.

Following their findings, Network  $B$  with separate columns yielded better results and had less trainable parameters. Thus, we use it in all our subsequent investigations. In particular, we identify two possible improvements: one related to the temperature used as an input, since this information is unavailable in simulators. The second area of improvement concerns the windowing size used for the force and position values used in the original paper, which was not sufficiently justified.

We use a similar split used by Ruppel et al. [10] to analyze these possible improvements. The test and validation sets consist of 30 chunks of data, each comprising 1000 data points. This configuration ensures that each set, independently,

accounts for 10% of the entire dataset. We train the network without early stopping for 50 epochs. We also conduct 10-fold cross-validation to enhance the statistical robustness of our metric reporting. We also do not consider the channels  $pac_1$  and  $tac$  in the metric calculation.

### A. Investigating the Training with Temperature Readings

The authors used the current temperature readings ( $tdc$ ) at timestep  $t = T$  of the BioTac sensor to predict the output values. Since the used simulation environment Gazebo [23] lacks temperature information, they suggest fixing that input value to a specific constant, specifically the mean temperature value of the entire collected dataset. However, the BioTac sensor is sensitive to temperature changes [13], and plotting the temperature values of the dataset, Figure 2, it can be seen that the temperature readings of the BioTac sensor continuously increase over time. This data suggests that fixing the temperature to a specific constant value will decrease the model’s performance.

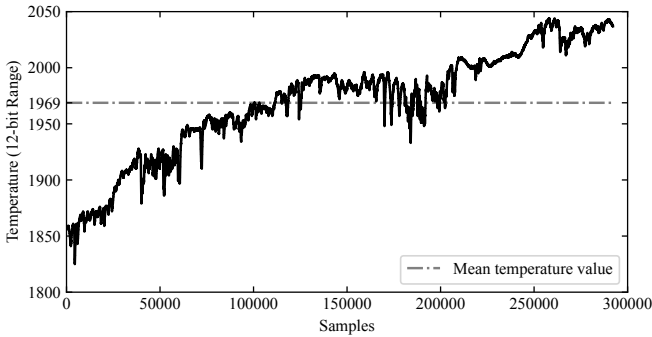


Fig. 2. Temperature values of the BioTac sensor in the Ruppel et al. [10]’s dataset. The dashed black line represents the average temperature value.

To test the effect of fixing the temperature value, we train Network  $B$  using the same hyperparameters used by the authors. However, during the testing phase, we first compute the network’s outputs using the correct temperature values provided in the dataset. Second, we compute the network’s output using a fixed temperature value. Here, we test all temperature values in the dataset, including the mean value suggested by Ruppel et al. [10]. The results are shown in Figure 3.

Examining Figure 3, we observe the lowest error when using the correct temperature as input, 9.6%. However, fixing the temperature in the best-case scenario leads to an averaged normalized MAE of 22.8%. Hence, fixing the input temperature value translates to a relative performance loss of 24.4%. We determine the relative loss between the two scenarios by determining the MAE of a naive model (53.7%), which always outputs the mean value of each channel as the prediction. These findings emphasize the need for an improved solution focusing on learning the data outputs without relying on temperature information since these are unavailable in the simulation environment.

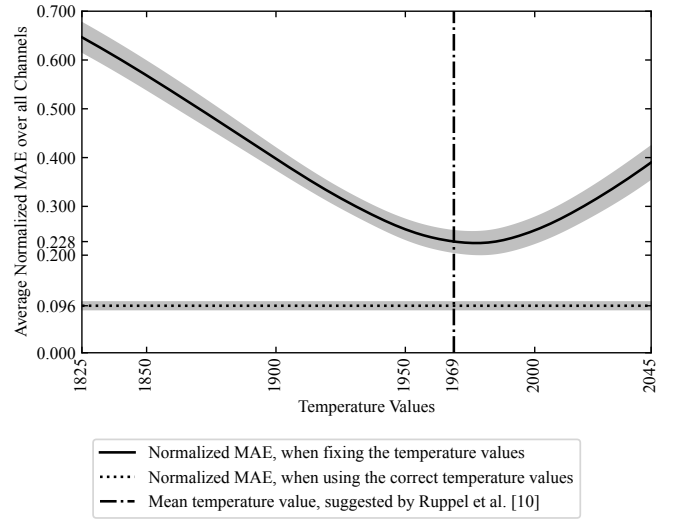


Fig. 3. Average normalized MAE over all channels for ten folds for Network  $B$ . The solid line represents the normalized MAE values when fixing the temperature input value and probing all temperatures in the dataset. The dashed line represents the normalized MAE, when using the appropriate correct temperature values of the test set. The shaded area represents the upper and lower limits calculated by the standard deviation. The vertical dash-dotted line represents the mean temperature of the entire dataset.

### B. Investigating the Windowing Size

The authors reported using the input vector force readings at three timesteps:  $T - 10$ ,  $T$ , and  $T + 10$ , omitting all intermediate timesteps and claiming their inclusion would lead to overfitting. Moreover, the use of force values from future timesteps, i.e.,  $T + 10$ , remains unexplained, and error reduction by including future values was not quantified. Thus, we conducted a thorough analysis of the choice of windowing size for force values and explored the impact of incorporating force values from future timesteps. To investigate this, we re-train Network  $B$  across six different conditions, keeping the current temperature and current position and varying in each experiment the windowing size of the force values as follows:

- 1) Forces  $F_t: \forall t \in \{T, T - 10, T + 10\}$ .
- 2) Forces  $F_t: \forall t \in \{T, T - 10\}$ .
- 3) Forces  $F_t: \forall t \in \{T\}$ .
- 4) Forces  $F_t: \forall t \in \{T, T - 10, T - 5, T + 5, T + 10\}$ .
- 5) Forces  $F_t: \forall t \in [T - 10, T + 10], t \in \mathbb{N}$ .
- 6) Forces  $F_t: \forall t \in [T - 10, T], t \in \mathbb{N}$ .

The training and validation loss curves for all conditions are depicted in Figure 4. Additionally, Table I illustrates these six conditions’ normalized mean absolute errors.

Figure 4 shows no sign of overfitting when using shorter intervals for the force values, as both training and validation loss curves converge over time. Table I reveals no apparent decrease in the metrics between all six input vector combinations when testing on a fixed temperature value equal to the mean temperature of the entire dataset. However, relying solely on the mean MAE over ten folds for comparison may be misleading. A more accurate approach is to compute paired differences of

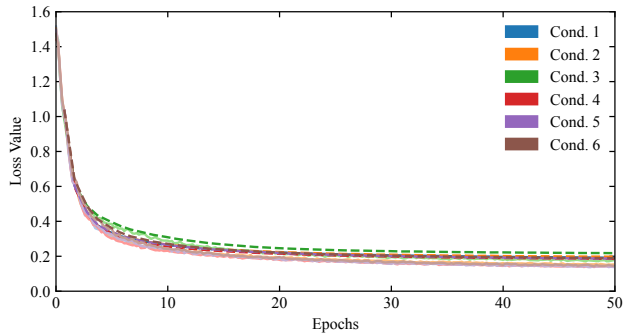


Fig. 4. Visualization of the training and validation loss values for all investigated conditions, showing no sign of overfitting. The solid line represents the mean training loss function over all ten folds. The dashed lines depict the mean validation loss over all ten folds.

TABLE I

RESULTS OF ALL TRAINED MODELS OVER ALL TEN FOLDS. THE VALUE IN PARENTHESES REPRESENTS THE STANDARD DEVIATION. MAE IS CALCULATED ON THE OUTPUT VALUES IN THE ORIGINAL SCALE. NORM. MAE IS CALCULATED ON THE NORMALIZED OUTPUT OF THE NETWORK; THE LOWER, THE BETTER.

	Nb. Param.	MAE	Norm. MAE	MAE	Norm. MAE
		Over all Channels		Electrodes Only	
1	806K	18.977 (1.719)	0.228 (0.022)	17.147 (1.642)	0.232 (0.023)
2	805K	23.220 (1.403)	0.237 (0.018)	18.941 (1.458)	0.239 (0.020)
3	804K	25.739 (1.539)	0.245 (0.019)	19.012 (1.498)	0.240 (0.021)
4	807K	21.944 (1.514)	0.233 (0.019)	19.003 (1.473)	0.240 (0.020)
5	819K	21.809 (1.548)	0.234 (0.020)	19.194 (1.529)	0.242 (0.021)
6	812K	22.319 (1.469)	0.234 (0.019)	19.038 (1.468)	0.240 (0.021)

normalized MAE for each fold separately and average these differences. This ensures a fair comparison, as the same ten folds are consistently used across all combination experiments.

We calculate the paired differences for the input combination used by the authors against all other investigated combinations. Statistical assessment is conducted using a left-tailed paired  $t$ -test. Since cross-validation with a single dataset involves reusing different training data points in multiple folds, it violates the independence assumptions of the paired  $t$ -test [32]. Hence, we employ the corrected paired  $t$ -test by Nadeau and Bengio [33], which accounts for the number of training and test data points and addresses the issue of high type I errors in the common paired  $t$ -test. The null hypothesis assumes that the paired difference is equal to 0, indicating no differences. The alternative hypothesis suggests that the normalized MAE of the first distribution is lower than that of the second distribution. We report the paired normalized MAE difference and the  $p$ -value in Table II.

Table II indicates that all comparisons are significant with

TABLE II

SIGNIFICANCE TEST WITH THE CORRECTED PAIRED  $t$ -TEST [33] CONDUCTED ON DIFFERENT INPUT COMBINATIONS PAIRS. THE FIRST VALUE DEPICTS THE PAIRED NORMALIZED MAE DIFFERENCE IN PERCENT OVER THE TEN FOLDS, AND THE SECOND VALUE BETWEEN PARENTHESES REPRESENTS THE  $p$ -VALUE.

	1 vs 2	1 vs 3	1 vs 4	1 vs 5	1 vs 6
Paired MAE Diff.	-0.893%	-1.763%	-0.513%	-0.630%	-0.655%
$p$ -value	(0.004)	(0.000)	(0.044)	(0.009)	(0.013)

$p < 0.05$ . These results show that adding force values from past and future timesteps, 1 vs 2 and 1 vs 3, decreases the error by ca. 0.9% and 1.8%, respectively. In addition, using shorter intervals of 10 ms, 1 vs 5, or including force values at timesteps  $T - 5$  and  $T + 5$ , 1 vs 4, does not reduce the error values. We will still consider these investigations and analyses and check different input combinations for our proposed solutions.

### III. METHODOLOGY

Based on the results presented in the previous section, we identify two areas of improvement for the BioTac simulation. The first concerns using or omitting temperature as an input value. The second concerns the windowing size used for the force and position values. This section describes the methodology used to optimize the BioTac simulation.

#### A. Training without Temperature Readings

Firstly, we train the baseline network  $B$  provided by Ruppel et al. [10] and use the mean temperature value of the dataset as input, and we use these scores as baseline. Next, we implement three approaches: a classical method using XGBoost [34], a gradient-boosting algorithm. XGBoost is fast and suitable for predicting continuous output variables [35]. A feed-forward deep neural network, given that this type of network is also commonly used in regression tasks [36]. Finally, we implement a transformer network since these proved beneficial in several time-series tasks [37]. Inspired by vision transformers (ViT) [38], we reformulate the classification task used in the ViT into a regression task by using a transformer encoder to predict the output vector. We present in Figure 5 the used backbone architecture for the transformer network [38].

We use Ruppel et al. [10]’s normalization, input, and output vectors with minor changes. For instance, we use the same input vector but without the temperature value as input resulting in an input vector of 12 values, i.e.,  $x$ ,  $y$ , and  $z$  at timestep  $T$ , force values ( $F_{xt}$ ,  $F_{yt}$ , and  $F_{zt}$ ) at timesteps  $T - 10$ ,  $T$ , and  $T + 10$ . The output value used consists of all 19 electrodes signals  $e_1 \dots e_{19}$ , the pressure values  $pac$ , and the dynamic vibration  $pac_0$  at timestep  $T$ , i.e., 21 output values. All channels’ input and output values are normalized using the z-score normalization.

In contrast to Ruppel et al., [10], we thoroughly fine-tune the hyperparameters of the tested models, namely XGBoost, feed-forward deep neural network, and transformer, using SMAC3 [39]. The normalized mean absolute error (MAE) is used as a cost function. We randomly split the data into training

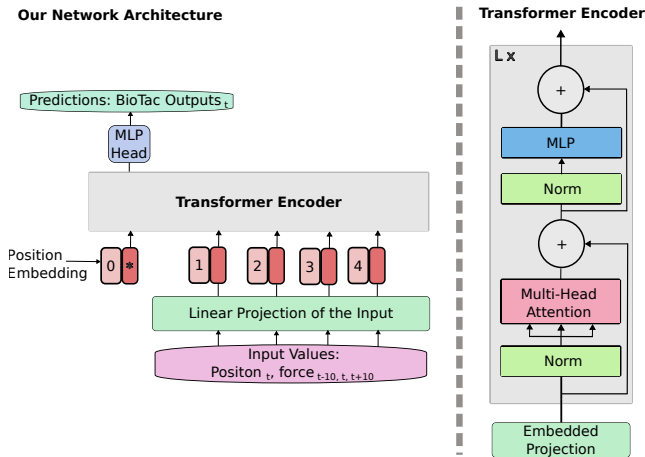


Fig. 5. Visualization of our used transformer architecture, based on Dosovitskiy et al. [38].  $L$  stands for the number of attention blocks used in the transformer encoder.

and validation sets (90% / 10%) with random seed values. To select the best configuration, we utilize hyperband [40] as an intensifier for the feed-forward network and the transformer, and we use a maximum budget of 30 epochs. The hyperparameters and their corresponding search space are reported in the supplementary material.

XGBoost does not support multi-variable output vectors. Thus, we fit one regressor for each channel. All regressors use the same hyperparameters determined with SMAC3. During training, we utilize the MAE loss function. The feed-forward network and the transformer are optimized using Adam [41], and we use the loss function presented in Equation 1.

Once the hyperparameters are determined, we use them to train our models from scratch. We split the dataset into training, validation, and test set in 80% / 10% / 10%, respectively. We perform early stopping on the validation loss to avoid overfitting. In contrast,  $k$ -fold cross-validation with ten folds is executed for a robust test set evaluation.

### B. Varying the Windowing Size

We also thoroughly investigate the choice of the input window for all tested approaches while omitting the temperature values. Beyond adjusting the force values sampling interval proposed in Section II-B, we explore two additional input combinations. For both input combinations 1 and 5, we include previous and next positional values within the sampling window to provide the network with contextual touch location information. These are respectively denoted as input combinations 7 and 8:

- 7) Current, last, and next position and forces  $F_t: \forall t \in \{T, T-10, T+10\}$ .
- 8) Current, last, and next position and forces  $F_t: \forall t \in [T-10, T+10], t \in \mathbb{N}$ .

We only test within the timestep range of  $t-10$  to  $t+10$  since the force values' sampling frequency of 100 Hz is a multiple of 10, which aligns with the position's sampling frequency of 10 Hz. We use the SMAC procedure described in the previous

Subsection III-A with the same hyperparameter search space. More details are in the supplementary material.

## IV. RESULTS

Here, we first report the results of our tested approaches that are trained without temperature. Next, we delve into a detailed investigation of a better input window.

### A. Training without Temperature Readings

After executing SMAC for all approaches, the models are trained, and the mean absolute error is calculated. We also determined the number of learnable parameters. For XGBoost regressors, we calculate the number of parameters by counting the number of nodes in all regressors across channels after training them. We then average this count over the folds, accounting for variability across each fold. The results are outlined in Table III.

TABLE III

RESULTS OF THE ALL TRAINED MODELS OVER ALL TEN FOLDS. THE VALUE IN PARENTHESES REPRESENTS THE STANDARD DEVIATION. THE METRICS ARE CALCULATED OVER ALL CHANNELS AND OVER ALL ELECTRODES. MAE IS CALCULATED ON THE OUTPUT VALUES IN THE ORIGINAL SCALE. NORM. MAE IS CALCULATED ON THE NORMALIZED OUTPUT OF THE NETWORK; THE LOWER, THE BETTER.

	Nb. Param.	MAE	Norm. MAE	MAE	Norm. MAE
		Over all Channels		Electrodes Only	
Ruppel et al. [10]	806K	18.977 (1.719)	0.228 (0.022)	17.147 (1.642)	0.232 (0.023)
Our XGBoost Regressor	1584K	<b>13.368</b> (1.340)	<b>0.150</b> (0.014)	<b>11.446</b> (1.204)	<b>0.150</b> (0.015)
Our Feed-Forward Neural Network	2233K	14.693 (1.386)	0.168 (0.015)	12.724 (1.252)	0.169 (0.016)
Our Transformer Encoder	<b>599K</b>	13.760 (1.400)	0.156 (0.016)	11.736 (1.280)	0.155 (0.016)

From Table III, it can be seen that the XGBoost and transformer approaches without temperature readings achieved a statistically significant improvement compared to the baseline. The absolute improvement measured in the normalized MAE ranges from 6.0% to 7.8%. The transformer encoder stands out for its compactness and reduced number of trainable parameters. While the XGBoost reaches the lowest normalized mean absolute error with 15.0%.

We also present the normalized MAE for each channel in Figure 6. There, it can be seen that the prediction errors differ substantially between channels. Particularly, electrodes 7, 8, 9, and 10, located at the tip of the BioTac sensor, result in the highest errors. This phenomenon is also seen for the XGBoost regressors, which have a dedicated regressor for each channel. This suggests a high non-linear dynamic of those electrodes.

Another possible reason for the differences in prediction errors could be the different number of samples available for each channel; i.e., the dataset might be unbalanced. We assessed that by selecting all data points at the beginning of a contact cycle having a force measurement higher than 0.3 N, located close to the BioTac sensor surface within a distance of less

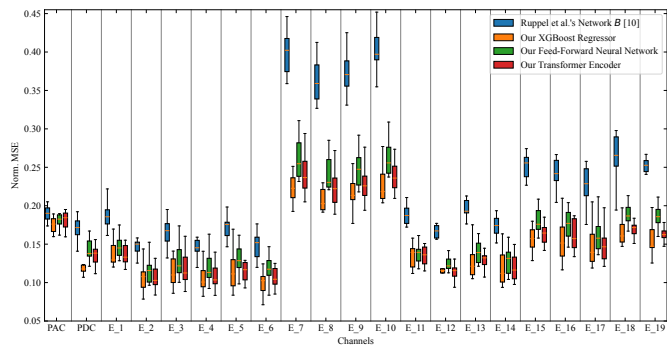


Fig. 6. Distribution of the normalized MAE for each channel and all models.

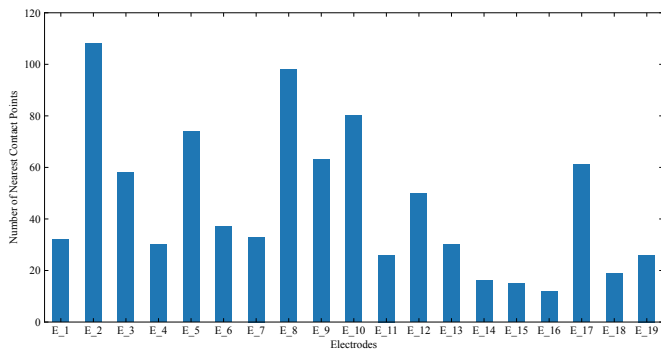


Fig. 7. Distribution of nearest contact points for each electrode.

than 2 *mm*. Subsequently, we identify the nearest electrode to each contact point using the 3D spatial positions of each electrode provided by Lin et al. [31]. The results are depicted in Figure 7.

Figure 7 highlights a non-uniform distribution of the number of touches near each electrode, implying an unbalanced dataset. However, this may not be the only reason contributing to the high MAE for specific electrodes. Based on the work of Lin et al. [31], we know that the BioTac sensor does not have radial symmetry. In addition, the fluid volume is not the same throughout the sensor. Particularly near electrodes 7, 8, 9, and 10, there is a higher fluid volume, which is partially visible in Figure 1(b). This supports the hypothesis that some of the error is due to the non-linear dynamics of the sensor.

### B. Varying the Windowing Size

After executing SMAC for all approaches, the models are trained, and the mean absolute error is calculated. The best hyperparameters for each trained model and input combination and the result metrics values are reported in the supplementary material.

Figure 8 presents the distribution of the normalized MAE for all combinations across all approaches. XGBoost attains the lowest MAE when trained with input combination 8, incorporating force values with shorter intervals and the last and next position, reaching a value of 14.8%, which represents an 8.0% improvement over the baseline. Likewise, our transformer

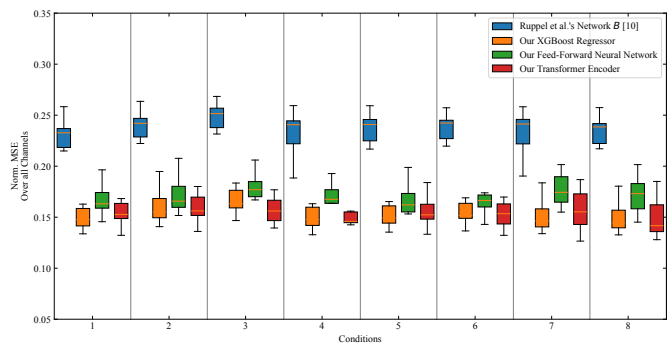


Fig. 8. The Normalized MAE was calculated over all channels for the four models and all input combinations.

encoder achieves a 7.8% improvement over the baseline with the same input combination.

We also conduct a corrected paired significance *t*-test [33] and report the paired normalized MAE difference and *p*-value for three specific comparison combinations: 5 vs 1, 7 vs 1 and 8 vs 1. The null hypothesis assumes that the paired difference is equal to 0, indicating no differences. The alternative hypothesis suggests that the normalized MAE of the first distribution is lower than that of the second distribution. Comparison 5 vs 1 assesses the impact of using force values of shorter intervals on error reduction, 7 vs 1 examines the effect of including the last and next different position, and 8 vs 1 evaluates the significance of incorporating more force values at shorter intervals along with the last and next position in error reduction. The paired normalized MAE difference, *t*-statistic and *p*-values for various combination pairs across all models are reported in the supplementary material.

Table IV indicates that adding more force values of shorter intervals does not directly lead to a reduction in error (5 vs 1), and the same applies when adding the last and next contact position (7 vs 1). However, combining both information yielded statistically significant improvements for the XGBoost regressor and the transformer encoder in the 8 vs 1 comparison. However, they are small, ranging between 0.2% and 0.6%.

TABLE IV  
SIGNIFICANCE TEST WITH THE CORRECTED PAIRED *t*-TEST [33] CONDUCTED ON DIFFERENT INPUT COMBINATIONS. THE FIRST VALUE DEPICTS THE PAIRED NORMALIZED MAE DIFFERENCE IN PERCENT OVER THE TEN FOLDS, AND THE SECOND VALUE BETWEEN PARENTHESES REPRESENTS THE *p*-VALUE.

	5 vs 1	7 vs 1	8 vs 1
Our XGBoost Regressor	0.282% (0.999)	-0.055% (0.190)	-0.208% <b>(0.009)</b>
Our Feed-Forward Neural Network	0.217% (0.715)	0.862% (0.961)	0.402% (0.758)
Our Transformer Encoder	0.077% (0.631)	0.188% (0.672)	-0.662% <b>(0.047)</b>

We also analyzed the significance between the tested approaches for all input combinations and reported the results. Table V reveals that both the XGBoost regressor and

TABLE V  
SIGNIFICANCE TEST WITH THE CORRECTED PAIRED  $t$ -TEST [33]  
CONDUCTED FOR ALL MODELS. THE FIRST VALUE DEPICTS THE PAIRED  
NORMALIZED MAE DIFFERENCE VALUE IN PERCENT, AND THE SECOND  
VALUE BETWEEN PARENTHESIS REPRESENTS THE  $p$ -VALUE.

vs	Our XGBoost Our FFNN	Our XGBoost Our Transformer	Our Transformer Our FFNN
1	-1.778% (0.000)	-0.531% (0.074)	-1.247% (0.002)
2	-1.360% (0.001)	-0.259% (0.250)	-1.101% (0.000)
3	-1.272% (0.003)	0.851% (0.996)	-2.123% (0.000)
4	-1.989% (0.000)	-0.032% (0.459)	-1.956% (0.000)
5	-1.279% (0.002)	-0.173% (0.322)	-1.107% (0.002)
6	-1.106% (0.003)	0.092% (0.616)	-1.197% (0.002)
7	-2.696% (0.000)	-0.774% (0.105)	-1.921% (0.000)
8	-2.389% (0.001)	-0.077% (0.439)	-2.312% (0.002)

the transformer encoder outperform the feed-forward neural network across all input combinations, demonstrating an error reduction ranging between 1.1% and 2.7%. However, a direct comparison between the XGBoost and transformer yields no significant values, suggesting no difference between the paired MAE.

Considering the importance of inference time during the deployment phase, we compute the inference time over 100 random inputs for all approaches and subsequently average the results. We run the inference calculation on a system with an AMD Ryzen 7 pro 4750u CPU. Figure 9 shows the inference time against the number of learnable parameters for all approaches and input combinations. For Ruppel et al. [10] baseline network, we only plot results for the first input combination, as all other combinations do not differ meaningfully in inference time or numbers of parameters.

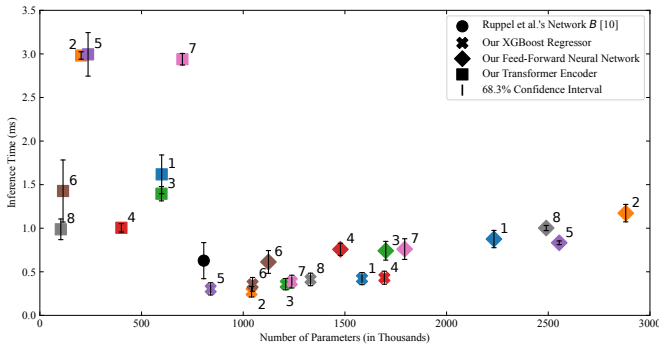


Fig. 9. Visualization of inference time against the number of parameters for each model over all input combinations.

Figure 9 reveals that the transformer network has the lowest

number of parameters compared to all other approaches, with an inference time ranging from 1.0  $ms$  up to 3.0  $ms$ , i.e., at least two to five times slower than the baseline. This is attributed to the quadratic time complexity of the attention blocks [42]. XGBoost is the fastest of the tested methods, with approximately half the inference time of the baseline network.

Considering that the number of floating-point operations per second (FLOPS) offers additional valuable insights into the energy consumption and inference efficiency of neural networks [43], we include the FLOPS count for all the neural networks, i.e., the baseline network, our feed-forward network, and our transformer encoder, in our supplementary material.

## V. CONCLUSION

We present optimizations to simulate tactile signals from the BioTac 2P sensor. Our contributions include a thorough analysis of Ruppel et al.'s work [10] and a thorough analysis of three alternative solutions. We focused on two areas: the use of temperature readings in the training pipeline of the network, and the choice of other variables in the input vector and the windowing.

We introduced three alternative models, i.e., an XGBoost regressor, a feed-forward neural network, and an adapted transformer encoder. Our results demonstrate that XGBoost and the transformer encoder can achieve significantly lower error values than the baseline. The absolute error reduction stands at 8.0%, while the relative error shows a statistically significant improvement of 14.9%. Furthermore, our investigations reveal a statistically significant improvement when incorporating force values from future timesteps and previous and next position values. Albeit, there are small between 0.2% up to 0.6% depending on the model used.

Notably, the XGBoost regressor has the lowest inference time, making it preferable for scenarios where simulation inference time is critical. Despite having less trainable parameters than all other models, the transformers exhibit the highest inference time of the tested models.

We also report on the limitations of the dataset. Particularly, the dataset is unbalanced and only includes a single indenter type. However, some of the errors can be attributed to the non-linear dynamics of the sensor, probably caused by the non-radial symmetry and non-uniform fluid volume throughout the sensor.

As future work, the dataset needs to be improved and extended to include different BioTac 2P sensors, varied surrounding temperatures, and several indenter shapes to enhance model robustness and generalizability, mitigating the absence of temperature information in the simulation environment and other non-linear dynamics.

Due to the differentiated performance of different electrodes due to non-linearities and unbalanced dataset, we also suggest training an ensemble of transformer networks to better deal with the non-linear dynamic of the sensor.

## REFERENCES

- [1] N. Navarro-Guerrero, S. Toprak, J. Josifovski, and L. Jamone, "Visuo-Haptic Object Perception for Robots: An Overview," *Autonomous Robots*, vol. 47, no. 4, pp. 377–403, 2023.
- [2] A. Parmiggiani, S. Dussioni, L. Natale, and G. Metta, "Tactile Sensors," in *Encyclopedia of Robotics*. Springer, 2020, pp. 1–11.
- [3] J. Josifovski, M. Malmir, N. Klarman, B. L. Zagar, N. Navarro-Guerrero, and A. Knoll, "Analysis of Randomization Effects on Sim2Real Transfer in Reinforcement Learning for Robotic Manipulation Tasks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 10 193–10 200.
- [4] S. Auddy, J. Hollenstein, M. Saveriano, A. Rodríguez-Sánchez, and J. Piater, "Continual Learning from Demonstration of Robotics Skills," *Robotics and Autonomous Systems*, vol. 165, p. 104427, 2023.
- [5] S. Wang, M. Lambeta, P.-W. Chou, and R. Calandra, "TACTO: A Fast, Flexible, and Open-Source Simulator for High-Resolution Vision-Based Tactile Sensors," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3930–3937, 2022.
- [6] M. Lambeta, P.-W. Chou, S. Tian, B. Yang, B. Maloon, V. R. Most, D. Stroud, R. Santos, A. Byagowi, G. Kammerer, D. Jayaraman, and R. Calandra, "DIGIT: A Novel Design for a Low-Cost Compact High-Resolution Tactile Sensor With Application to In-Hand Manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 3838–3845, 2020.
- [7] A. Padmanabha, F. Ebert, S. Tian, R. Calandra, C. Finn, and S. Levine, "OmniTact: A Multi-Directional High-Resolution Touch Sensor," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 618–624.
- [8] A. Melnik, L. Lach, M. Plappert, T. Korthals, R. Haschke, and H. Ritter, "Using Tactile Sensing to Improve the Sample Efficiency and Performance of Deep Deterministic Policy Gradients for Simulated in-Hand Manipulation Tasks," *Frontiers in Robotics and AI*, vol. 8, no. 538773, 2021.
- [9] J. Geukes, M. Nakatenus and R. Calandra. "Gazebo plugin for the icub skin," 2017. Online Available: <https://github.com/robertocalandra/icub-gazebo-skin>.
- [10] P. Ruppel, Y. Jonetzko, M. Görner, N. Hendrich, and J. Zhang, "Simulation of the SynTouch BioTac Sensor," in *International Conference on Intelligent Autonomous Systems (IAS)*, ser. AISC, vol. 867. Springer International Publishing, 2019, pp. 374–387.
- [11] Y. S. Narang, B. Sundaralingam, K. Van Wyk, A. Mousavian, and D. Fox, "Interpreting and Predicting Tactile Signals for the SynTouch BioTac," *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1467–1487, 2021.
- [12] B. S. Zapata-Impata, P. Gil, Y. Mezouar, and F. Torres, "Generation of Tactile Data From 3D Vision and Target Robotic Grasps," *IEEE Transactions on Haptics*, vol. 14, no. 1, pp. 57–67, 2021.
- [13] N. Wettels, D. Popovic, V. J. Santos, R. S. Johansson, and G. E. Loeb, "Biomimetic Tactile Sensor for Control of Grip," in *IEEE International Conference on Rehabilitation Robotics (ICORR)*, 2007, pp. 923–932.
- [14] N. Wettels, J. A. Fishel, and G. E. Loeb, "Multimodal Tactile Sensor," in *The Human Hand as an Inspiration for Robot Hand Development*, ser. Springer Tracts in Advanced Robotics. Springer International Publishing, 2014, no. 95, pp. 405–429.
- [15] L. Natale and G. Cannata, "Tactile Sensing," in *Humanoid Robotics: A Reference*. Springer Netherlands, 2019, pp. 2539–2561.
- [16] Y. Chebotar, K. Hausman, Z. Su, G. Sukhatme, and S. Schaal, "Self-Supervised Regrasping Using Spatio-Temporal Tactile Features and Reinforcement Learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1960–1966.
- [17] D. Xu, G. E. Loeb, and J. A. Fishel, "Tactile Identification of Objects Using Bayesian Exploration," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 3056–3061.
- [18] M. S. Arian, C. A. Blaine, G. E. Loeb, and J. A. Fishel, "Using the BioTac as a Tumor Localization Tool," in *IEEE Haptics Symposium (HAPTICS)*, 2014, pp. 443–448.
- [19] C. Pacchierotti, D. Prattichizzo, and K. J. Kuchenbecker, "Cutaneous Feedback of Fingertip Deformation and Vibration for Palpation in Robotic Surgery," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 2, pp. 278–287, 2016.
- [20] SynTouch, BioTac® Product Manual, August, 2018, V21: <https://www.syntouchinc.com/wp-content/uploads/2018/08/BioTac-Manual-V.21.pdf>.
- [21] The Shadow Dexterous Hand: <https://www.shadowrobot.com/dexterous-hand-series/>.
- [22] ATI Industrial Automation Inc.: F/T Sensor Nano17 [https://www.ati-ia.com/products/ft/ft\\_models.aspx?id=Nano17](https://www.ati-ia.com/products/ft/ft_models.aspx?id=Nano17).
- [23] N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2004, pp. 2149–2154.
- [24] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *International Conference on Neural Information Processing Systems (NIPS)*, ser. NIPS'17. Curran Associates Inc., 2017, pp. 5105–5114.
- [25] NVIDIA Isaac Gym Simulator: <https://developer.nvidia.com/isaac-gym>.
- [26] Y. Narang, B. Sundaralingam, M. Macklin, A. Mousavian, and D. Fox, "Sim-to-Real for Robotic Tactile Sensing Via Physics-Based Simulation and Learned Latent Projections," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6444–6451.
- [27] T. Taunyazov, L. S. Song, E. Lim, H. H. See, D. Lee, B. C. Tee, and H. Soh, "Extended Tactile Perception: Vibration Sensing through Tools and Grasped Objects," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1755–1762.
- [28] SynTouch, BioTac® SP Product Manual, August, 2018, V3: <https://www.syntouchinc.com/wp-content/uploads/2018/08/BioTac-SP-Manual-V.3.pdf>.
- [29] G. Olmschenk, Z. Zhu, and H. Tang, "Generalizing Semi-Supervised Generative Adversarial Networks to Regression Using Feature Contrast-ing," *Computer Vision and Image Understanding*, vol. 186, pp. 1–12, 2019.
- [30] E. Olson, "AprilTag: A Robust and Flexible Visual Fiducial System," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3400–3407.
- [31] C.-H. Lin, J. A. Fishel, and G. E. Loeb, "Estimating point of contact, force and torque in a biomimetic tactile sensor with deformable skin," in "Technical report, SynTouch LLC", 2013.
- [32] T. G. Dietterich, "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms," *Neural Computation*, vol. 10, no. 7, pp. 1895–1923, Oct. 1998.
- [33] C. Nadeau and Y. Bengio, "Inference for the Generalization Error," *Machine Learning*, vol. 52, no. 3, pp. 239–281, Sep. 2003.
- [34] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, ser. KDD '16. ACM, 2016, pp. 785–794.
- [35] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A Comparative Analysis of Gradient Boosting Algorithms," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1937–1967, 2021.
- [36] L. Das, A. Sivaram, and V. Venkatasubramanian, "Hidden Representations in Deep Neural Networks: Part 2. Regression Problems," *Computers & Chemical Engineering*, vol. 139, p. 106895, 2020.
- [37] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in Time Series: A Survey," in *International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 6, 2023, pp. 6778–6786.
- [38] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *International Conference on Learning Representations (ICLR)*, 2021.
- [39] M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, T. Ruhkopf, R. Sass, and F. Hutter, "SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization," *Journal of Machine Learning Research*, vol. 23, no. 54, pp. 1–9, 2022.
- [40] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization," *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2018.
- [41] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations (ICLR)*, ser. 3rd, 2015, p. 15.
- [42] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient Transformers: A Survey," *ACM Computing Surveys*, vol. 55, no. 6, pp. 109:1–109:28, 2022.
- [43] R. Tang, W. Wang, Z. Tu, and J. Lin, "An Experimental Analysis of the Power Consumption of Convolutional Neural Networks for Keyword Spotting," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, AB, Canada, 2018, pp. 5479–5483.



TABLE VI  
SUMMARY OF THE HYPERPARAMETER SEARCH SPACE USED FOR THE XGBOOST REGRESSOR, WHEN RUNNING SMAC3 [39] AND THE SELECTED CONFIGURATION FOR ALL INPUT COMBINATIONS.

Hyperparameters	Our XGBoost Regressor								
	Search Space	Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7	Input 8
eta	Uniform Float $\in [0.0001, 0.5]$	0.0431	0.06818	0.04942	0.04457	0.07178	0.06647	0.04901	0.04798
gamma	Uniform Integer $\in [0, 10]$	1	7	2	1	7	3	3	2
number estimate	Uniform Integer $\in [100, 1000]$	972	155	932	880	230	913	715	949
max depth	Uniform Integer $\in [1, 10]$	10	10	10	10	10	10	10	10
min child weight	Uniform Integer $\in [1, 100]$	95	6	87	67	59	83	84	84
max delta step	Uniform Integer $\in [0, 10]$	7	10	0	1	5	3	10	1
subsample	Uniform Float $\in [0.5, 1]$	0.647	0.9632	0.5042	0.5068	0.6114	0.5486	0.507	0.5038
colsample bytree	Uniform Float $\in [0.5, 1]$	0.9825	0.9517	0.8671	0.868	0.7449	0.8242	0.9927	0.7814
colsample bylevel	Uniform Float $\in [0.5, 1]$	0.9819	0.9223	0.8572	0.839	0.9264	0.7642	0.9479	0.8407
colsample bynode	Uniform Float $\in [0.5, 1]$	0.8042	0.9155	0.6173	0.7416	0.9802	0.924	0.8621	0.9989

TABLE VII  
TABLE SUMMARIZING THE HYPERPARAMETER SEARCH SPACE USED FOR THE FEED-FORWARD DEEP NEURAL NETWORK, WHEN RUNNING SMAC3 [39] AND THE SELECTED CONFIGURATION FOR ALL INPUT COMBINATIONS.

Hyperparameters	Our Feed-Forward Neural Network								
	Search Space	Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7	Input 8
Batch Size	Categorical [256, 512]	256	512	256	512	512	512	512	512
Learning Rate	Categorical $a \times e^{-c}$ for $a \in \mathbb{N}^+$ and $\in [1, 9]$ $c \in \mathbb{N}^+$ and $\in [2, 5]$	0.0003	0.0002	0.0005	0.0006	0.0003	0.0006	0.0004	0.0005
Number of Layers ( $L$ )	Uniform Int Lower: 4 Upper: 12	7	10	7	7	7	9	7	8
Number of Neurons in Layer $i$ for $i \in [0, L]$	Uniform Int Lower: 50 Upper: 1000 Step: 10	[860, 670, 160, 580, 900, 1000, 440]	[590,300, 820,520, 90,670, 850,120, 330,570]	[620, 470, 120, 620, 830, 890, 350]	[730, 390, 120, 630, 770, 720, 380]	[820,740, 190,740, 1000,850, 430]	[580,710, 600,170, 270,350, 70,780, 690]	[580,580, 160,450, 920,920, 380]	[470,600, 440,830, 790,900, 190,270]
Activation function in Layer $i$ for $i \in [0, L]$	Categorical [sigmoid, relu, hardtanh, tanh, leakyrelu, elu]	[hardtanh, tanh, elu, relu, leakyrelu, relu, leakyrelu]	[elu, relu, hardtanh, elu, leakyrelu, relu, elu, elu, elu, leakyrelu]	[tanh, elu, elu, leakyrelu, leakyrelu, elu, leakyrelu]	[tanh, tanh, elu, hardtanh, sigmoid, leakyrelu, leakyrelu]	[tanh, relu, elu, leakyrelu, elu, relu, leakyrelu]	[tanh, relu, leakyrelu, elu, leakyrelu, leakyrelu, hardtanh, leakyrelu]	[tanh, relu, leakyrelu, hardtanh, sigmoid, relu, leakyrelu]	[tanh, relu, elu, leakyrelu, hardtanh, leakyrelu, elu, leakyrelu]
Negative Slope for Leakyrelu	Categorical $a \times e^{-1}$ for $a \in \mathbb{N}^+$ and $\in [1, 9]$	0.5	0.7	0.4	0.2	0.1	0.4	0.3	0.6

TABLE VIII

TABLE SUMMARIZING THE HYPERPARAMETER SEARCH SPACE USED FOR THE TRANSFORMER ENCODER, WHEN RUNNING SMAC3 [39] AND THE SELECTED CONFIGURATION FOR ALL INPUT COMBINATIONS.

Hyperparameters	Our Transformer Encoder								
	Search Space	Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7	Input 8
Batch Size	Categorical [256, 512]	512	512	512	256	512	256	512	256
Learning Rate	Categorical $a \times e^{-c}$ for $a \in \mathbb{N}^+$ and $\in [1, 9]$ $c \in \mathbb{N}^+$ and $\in [2, 5]$	0.00009	0.00004	0.00009	0.0004	0.004	0.0003	0.00004	0.002
Number of Layers ( $L$ )	Uniform Int Lower: 2 Upper: 8	3	8	3	2	7	4	7	2
Number of Multi-Heads	Categorical [1, 2, 4, 8]	4	8	4	8	1	4	1	8
Dropout Rate	Categorical [0.0, 0.1, 0.2, 0.3, 0.4, 0.5]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Embedding Dimensions	Categorical [32, 64, 128, 256, 512, 1024]	128	128	128	128	64	32	128	64
Hidden Dimension	Categorical [64, 128, 256, 512, 1024]	512	512	512	512	128	256	128	256

TABLE IX

RESULTS OF ALL TRAINED NETWORKS OVER ALL 10 FOLDS FOR ALL INPUT COMBINATIONS. THE VALUE IN PARENTHESES REPRESENTS THE STANDARD DEVIATION. THE METRICS ARE CALCULATED OVER ALL CHANNELS AND OVER ALL ELECTRODES. MAE IS CALCULATED ON THE OUTPUT VALUES IN THE ORIGINAL SCALE. NORM. MAE IS CALCULATED ON THE NORMALIZED OUTPUT OF THE NETWORK; THE LOWER, THE BETTER.

	Nb. Param.	MSE	Norm. MSE	MSE	Norm. MSE	
		Over all Channels	Channels	Electrodes Only	Electrodes Only	
Ruppel et al. [10]	1	806K	18.977 (1.719)	0.228 (0.022)	17.147 (1.642)	0.232 (0.023)
	2	805K	23.220 (1.403)	0.237 (0.018)	18.941 (1.458)	0.239 (0.020)
	3	804K	25.739 (1.539)	0.245 (0.019)	19.012 (1.498)	0.240 (0.021)
	4	807K	21.944 (1.514)	0.233 (0.019)	19.003 (1.473)	0.240 (0.020)
	5	819K	21.809 (1.548)	0.234 (0.020)	19.194 (1.529)	0.242 (0.021)
	6	812K	22.319 (1.469)	0.234 (0.019)	19.038 (1.468)	0.240 (0.021)
	7	809K	22.497 (1.466)	0.234 (0.019)	18.811 (1.449)	0.238 (0.020)
	8	822K	21.555 (1.504)	0.231 (0.019)	18.927 (1.489)	0.239 (0.020)
Our XGBoost Regressor	1	1584K	13.368 (1.340)	0.150 (0.015)	11.446 (1.204)	<b>0.150</b> (0.015)
	2	1041K	14.372 (1.446)	0.159 (0.016)	12.002 (1.298)	0.158 (0.016)
	3	1209K	15.994 (1.622)	0.168 (0.017)	12.373 (1.340)	0.163 (0.017)
	4	1694K	13.275 (1.430)	0.152 (0.016)	11.638 (1.335)	0.153 (0.016)
	5	840K	13.407 (1.393)	0.153 (0.015)	11.774 (1.277)	0.154 (0.016)
	6	1046K	13.785 (1.385)	0.156 (0.015)	11.866 (1.277)	0.156 (0.016)
	7	1239K	13.299 (1.330)	0.150 (0.015)	11.375 (1.182)	<b>0.150</b> (0.015)
	8	1331K	<b>12.873</b> (1.255)	<b>0.148</b> (0.014)	11.385 (1.156)	<b>0.150</b> (0.014)

	Nb. Param.	MSE	Norm. MSE	MSE	Norm. MSE	
		Over all Channels	Channels	Electrodes Only	Electrodes Only	
Our Feed-Forward Neural Network	1	2233K	14.693 (1.386)	0.168 (0.015)	12.724 (1.252)	0.169 (0.016)
	2	2881K	15.624 (1.536)	0.173 (0.017)	12.965 (1.388)	0.171 (0.018)
	3	1701K	17.010 (1.314)	0.181 (0.014)	13.265 (1.105)	0.175 (0.014)
	4	1478K	14.712 (1.317)	0.171 (0.015)	13.109 (1.247)	0.173 (0.016)
	5	2554K	14.056 (1.213)	0.166 (0.013)	12.721 (1.100)	0.169 (0.014)
	6	1124K	14.476 (1.150)	0.167 (0.013)	12.676 (1.043)	0.168 (0.013)
	7	1794K	15.231 (1.344)	0.177 (0.016)	13.351 (1.242)	0.178 (0.016)
	8	2490K	14.517 (1.450)	0.172 (0.017)	13.204 (1.404)	0.175 (0.017)
Our Transformer Encoder	1	599K	13.760 (1.400)	0.156 (0.016)	11.736 (1.280)	0.155 (0.016)
	2	203K	14.534 (1.482)	0.162 (0.017)	12.102 (1.374)	0.160 (0.017)
	3	598K	15.229 (1.617)	0.160 (0.017)	11.595 (1.328)	0.154 (0.017)
	4	401K	13.204 (1.363)	0.152 (0.016)	11.564 (1.266)	0.153 (0.016)
	5	237K	13.441 (1.241)	0.155 (0.014)	11.824 (1.147)	0.156 (0.015)
	6	114K	13.669 (1.505)	0.155 (0.018)	11.719 (1.376)	0.155 (0.018)
	7	701K	13.927 (1.614)	0.158 (0.019)	11.838 (1.468)	0.157 (0.019)
	8	<b>103K</b>	12.984 (1.552)	0.149 (0.018)	<b>11.334</b> (1.473)	<b>0.150</b> (0.019)

TABLE X

EXTENDED SIGNIFICANCE TEST WITH THE CORRECTED PAIRED  $t$ -TEST [33] CONDUCTED ON DIFFERENT INPUT COMBINATIONS FOR ALL NETWORKS. THE FIRST VALUE DEPICTS THE PAIRED NORMALIZED MAE DIFFERENCE IN PERCENT OVER THE TEN FOLDS, THE SECOND VALUE REPRESENTS  $t$ -STATISTIC, AND THE THIRD VALUE BETWEEN PARENTHESIS REPRESENTS THE  $p$ -VALUE.

	1 vs 2	1 vs 3	1 vs 4	1 vs 5	1 vs 6	7 vs 1	8 vs 1	5 vs 6	8 vs 5
Ruppel et al. [10]	-0.893% <b>(0.004)</b>	-1.763% <b>(0.000)</b>	-0.513% <b>(0.044)</b>	-0.630% <b>(0.009)</b>	-0.655% <b>(0.013)</b>	0.578% (0.967)	0.353% (0.915)	-0.025% (0.398)	-0.277% <b>(0.007)</b>
Our XGBoost Regressor	-0.859% <b>(0.000)</b>	-1.773% <b>(0.000)</b>	-0.113% (0.129)	-0.282% <b>(0.001)</b>	-0.539% <b>(0.000)</b>	-0.055% (0.190)	-0.208% <b>(0.009)</b>	-0.258% <b>(0.009)</b>	-0.490% <b>(0.001)</b>
Our Feed-Forward Neural Network	-0.441% -2.222 <b>(0.027)</b>	-1.267% -5.436 <b>(0.000)</b>	-0.324% -2.184 <b>(0.028)</b>	0.217% 0.589 (0.715)	0.133% 0.373 (0.641)	0.862% 1.995 (0.961)	0.402% 0.729 (0.758)	-0.084% -0.226 (0.413)	0.620% 1.296 (0.886)
Our Transformer Encoder	-0.587% -2.479 <b>(0.018)</b>	-0.391% -1.400 (0.097)	0.385% 1.074 (0.845)	0.077% 0.344 (0.631)	0.083% 0.438 (0.664)	0.188% 0.461 (0.672)	-0.662% -1.870 <b>(0.047)</b>	0.007% 0.019 (0.507)	-0.585% -1.537 (0.079)

TABLE XI

EXTENDED SIGNIFICANCE TEST WITH THE CORRECTED PAIRED  $t$ -TEST [33] CONDUCTED FOR ALL NETWORK PAIRS. THE FIRST VALUE DEPICTS THE PAIRED NORMALIZED MAE DIFFERENCE IN PERCENT OVER THE TEN FOLDS, THE SECOND VALUE REPRESENTS  $t$ -STATISTIC, AND THE THIRD VALUE BETWEEN PARENTHESIS REPRESENTS THE  $p$ -VALUE.

vs	Our XGBoost Ruppel et al. [10]	Our FFNN Ruppel et al. [10]	Our Transformer Ruppel et al. [10]	Our XGBoost Our FFNN	Our XGBoost Our Transformer	Our Transformer Our FFNN
1	-7.740% -9.785 <b>(0.000)</b>	-5.962% -7.741 <b>(0.000)</b>	-7.209% -8.670 <b>(0.000)</b>	-1.778% -5.188 <b>(0.000)</b>	-0.531% -1.581 (0.074)	-1.247% -3.690 <b>(0.002)</b>
2	-7.774% -9.962 <b>(0.000)</b>	-6.414% -8.136 <b>(0.000)</b>	-7.515% -8.956 <b>(0.000)</b>	-1.360% -4.284 <b>(0.001)</b>	-0.259% -0.704 (0.250)	-1.101% -5.338 <b>(0.000)</b>
3	-7.730% -9.936 <b>(0.000)</b>	-6.457% -8.509 <b>(0.000)</b>	-8.581% -11.401 <b>(0.000)</b>	-1.272% -3.623 <b>(0.003)</b>	0.851% 3.407 (0.996)	-2.123% -6.570 <b>(0.000)</b>
4	-8.140% -9.546 <b>(0.000)</b>	-6.151% -7.352 <b>(0.000)</b>	-8.108% -8.958 <b>(0.000)</b>	-1.989% -5.653 <b>(0.000)</b>	-0.032% -0.106 (0.459)	-1.956% -11.260 <b>(0.000)</b>
5	-8.089% -9.960 <b>(0.000)</b>	-6.809% -8.364 <b>(0.000)</b>	-7.916% -10.944 <b>(0.000)</b>	-1.279% -3.888 <b>(0.002)</b>	-0.173% -0.477 (0.322)	-1.107% -3.810 <b>(0.002)</b>
6	-7.856% -10.032 <b>(0.000)</b>	-6.750% -7.725 <b>(0.000)</b>	-7.947% -9.859 <b>(0.000)</b>	-1.106% -3.610 <b>(0.003)</b>	0.092% 0.303 (0.616)	-1.197% -3.918 <b>(0.002)</b>
7	-8.373% -10.568 <b>(0.000)</b>	-5.678% -7.579 <b>(0.000)</b>	-7.599% -7.886 <b>(0.000)</b>	-2.696% -5.235 <b>(0.000)</b>	-0.774% -1.347 (0.105)	-1.921% -5.899 <b>(0.000)</b>
8	-8.301% -10.363 <b>(0.000)</b>	-5.912% -5.268 <b>(0.000)</b>	-8.224% -9.773 <b>(0.000)</b>	-2.389% -4.242 <b>(0.001)</b>	-0.077% -0.157 (0.439)	-2.312% -4.004 <b>(0.002)</b>

TABLE XII

NUMBER OF PARAMETERS IN THOUSANDS, INFERENCE TIME IN MILLISECONDS, AND FLOATING-POINT OPERATIONS PER SECOND (FLOPS) IN MILLIONS FOR ALL APPROACHES ACROSS ALL INPUT COMBINATIONS. THE NUMBER OF FLOPS IS ONLY CALCULATED FOR THE NEURAL NETWORKS.

	Ruppel et al.'s Network $B$ [10]			Our XGBoost Regressor			Our FF Neural Network			Our Transformer Encoder		
	Numb. Param.	Inference ( $ms$ )	Numb. FLOPS	Numb. Param.	Inference ( $ms$ )	Numb. FLOPS	Numb. Param.	Inference ( $ms$ )	Numb. FLOPS	Numb. Param.	Inference ( $ms$ )	Numb. FLOPS
1	806K	0.628	1.61M	1584K	0.422	-	2233K	0.876	4.46M	599K	1.618	5.97M
2	805K	0.539	1.61M	1041K	0.272	-	2881K	1.173	5.75M	203K	2.982	1.63M
3	804K	0.539	1.61M	1209K	0.358	-	1701K	0.741	3.40M	598K	1.396	3.58M
4	807K	0.555	1.61M	1694K	0.432	-	1478K	0.757	2.95M	401K	1.005	5.59M
5	819K	0.572	1.65M	840K	0.304	-	2554K	0.833	5.10M	237K	2.995	11.64M
6	812K	0.552	1.62M	1046K	0.356	-	1124K	0.613	2.24M	114K	1.427	2.98M
7	809K	0.572	1.62M	1239K	0.387	-	1794K	0.760	3.58M	701K	2.939	9.89M
8	822K	0.607	1.65M	1331K	0.412	-	2490K	1.002	4.98M	103K	0.988	5.30M