Graziano Alfredo Manduzio[1], Federico Galatolo[1], Mario G C A Cimino[1], Mattia Bruscia[1], Lorenzo Cominelli[1], and Enzo Pasquale Scilingo[1]

[1]Affiliation not available

May 02, 2024

Abstract

In recent years, the robotics field has witnessed an unprecedented surge in the development of humanoid robots, which bear an increasingly close resemblance to human beings in appearance and functionality. This evolution has presented researchers with complex challenges, particularly in the domain of controlling the increasing number of robotic motors that animate these lifelike figures. This paper focuses on a novel approach to managing the intricate facial expressions of a humanoid face endowed with 22 degrees of freedom. We introduce a groundbreaking inverse kinematic model that leverages deep learning regression techniques to bridge the gap between the visual representation of human facial expressions and the corresponding servo motor configurations required to replicate these expressions. By mapping image space to servo motor space, our model enables precise, dynamic control over facial expressions, enhancing the robot's ability to engage in more nuanced and human-like interactions. Our methodology not only addresses the technical complexities associated with the fine-tuned control of facial motor servos but also contributes to the broader discourse on improving humanoid robots' social adaptability and interaction capabilities. Through extensive experimentation and validation, we demonstrate the efficacy and robustness of our approach, marking a significant advancement in humanoid robotics control systems.

Advanced Control of Humanoid Facial Robotics: A Deep Learning Approach to Inverse Kinematics

Graziano A. Manduzio, Federico Galatolo, Mario G. C. A. Cimino, Mattia Bruscia, Lorenzo Cominelli*, and Enzo Pasquale Scilingo*

***Abstract*—In recent years, the robotics field has witnessed an unprecedented surge in the development of humanoid robots, which bear an increasingly close resemblance to human beings in appearance and functionality. This evolution has presented researchers with complex challenges, particularly in the domain of controlling the increasing number of robotic motors that animate these lifelike figures. This paper focuses on a novel approach to managing the intricate facial expressions of a humanoid face endowed with 22 degrees of freedom. We introduce a groundbreaking inverse kinematic model that leverages deep learning regression techniques to bridge the gap between the visual representation of human facial expressions and the corresponding servo motor configurations required to replicate these expressions. By mapping image space to servo motor space, our model enables precise, dynamic control over facial expressions, enhancing the robot's ability to engage in more nuanced and human-like interactions. Our methodology not only addresses the technical complexities associated with the fine-tuned control of facial motor servos but also contributes to the broader discourse on improving humanoid robots' social adaptability and interaction capabilities. Through extensive experimentation and validation, we demonstrate the efficacy and robustness of our approach, marking a significant advancement in humanoid robotics control systems.**

***Index Terms*—deep learning, inverse kinematics, IK, human-robot interaction, HRI, facial robotics, social robotics, face detection, face recognition, facial expression imitation, FEI, MTCNN, InceptionResnetV1, hyperparameter optimization.**

I. Introduction

THE development of humanoid robots has increasingly focused on achieving human-like appearances and functionalities, presenting significant challenges in the control of robotic motors for nuanced movements. *Inverse kinematics* (IK) is a critical area in this regard, facilitating the translation of desired end-effector positions into the necessary joint configurations for movement. The complexity of this task escalates with the number of degrees of freedom in the robot, making IK a central focus in robotic research [23, 2]. The integration of machine learning into robotic control systems for solving IK problems represents a significant advancement in the field. Choi et al. [5], and Daya et al. [7], both found that neural networks can effectively model manipulator inverse kinematics, with Daya specifically proposing a neural network architecture for this purpose. Aggogeri et al., further enhanced this approach by using a sequential procedure and a genetic algorithm, resulting in improved accuracy and reduced manual settings [1]. Kuroe et al., introduced a new method that simultaneously represents the relations of positions and velocities, leading to more accurate solutions [12]. Similarly, the approach of Lu et al., offers a more efficient and flexible solution for determining joint positions from desired end-effector locations [15]. *Deep Reinforcement Learning* (DRL), in particular, offers a promising approach by providing models that can learn efficient and practical solutions for controlling robotic movements, including the nuanced motions required for facial expressions. This approach not only enhances the robot's ability to perform complex tasks but also its capacity for natural and engaging human interaction (Malik et al., [17]). In recent years, the advent of robots that closely mimic human appearance and behavior has further complicated the challenges associated with inverse kinematics, particularly in the realm of facial robotics. Advanced robotic faces now feature skin-like materials, mimicking the elasticity and texture of human skin, [16]. This development of *Facial Expression Imitation* (FEI) robotic skills, has necessitated sophisticated inverse kinematics solutions to accurately replicate the complex range of human facial expressions. Recent research efforts have focused on enhancing humanoid robots' ability to mimic human emotional expressions through advanced interaction models. Park et al., achieved a variety of facial expressions by adjusting dynamics and enhancing the realism of a robot through the integration of secondary actions, such as physiological movements like blinking and sinusoidal movements associated with breathing [18]. They utilized a second-order differential equation derived from the linear affective space-expression model to generate dynamic expression movements. Breazeal at al., elicited Kismet's robot emotions through the interpolation within a three-dimensional space, where each dimension represents valence, arousal, and position [4]. As the affective state progresses towards extreme values within this space, expressions intensify accordingly. In the last few year, deterministic approaches are giving way because their limitation of representing detailed facial expressions in favor of more advanced stochastic-based techniques. The ability to reproduce a huge variety of facial expression is important to enhance emphatic behaviour in social robots. In this respect, machine learning techniques are taking the lead. Wu et al., utilized correlations between *Action Units* (AUs) and servos within a highly articulated robot face to establish a linear mapping between the two [24]. This study delves into the process of self-guided learning aimed at achieving realistic facial expression

G. A. Manduzio, F. Galatolo, M. G. C. A. Cimino, M. Bruscia, L. Cominelli and E. P. Scilingo are with the Dipartimento di Ingegneria dell'Informazione, Università di Pisa, 56122 Pisa, Italy (e-mail: grazianoalfredo.manduzio@phd.unipi.it), (e-mail: {federico.galatolo, mario.cimino, lorenzo.cominelli, enzo.scilingo}@unipi.it), (e-mail: mattia.bruscia@studenti.unipi.it).

production by a robotic head named Einstein, which features 31 degrees of freedom. Facial motor parameters are learned using feedback from real-time facial expression recognition from video. The experiments demonstrate that the mapping of servos to expressions is successfully learned in less than an hour of training time. Huang et al., attempted to develop a real-time expression mimicking method for humanoid robots using deep *Long Short-Term Memory* (LSTM) networks [21]. They proposed a multi-frame imitation algorithm that integrates an inverse mechanical model and a motion trend model. In another study, Huang et al., proposed a mapping system from facial feature sequences to motor position sequences based on *smooth-constraint reversed mechanical model* (SRMM) by combining a sequence-to-sequence deep learning model (a multi-layer encoding-decoding LSTM structure was used) in addition to a loss function that incorporates velocity and acceleration constraints, thereby improving the smoothness of the resulting motor position sequences [9]. Particularly in medical settings, the ability of robots to establish empathetic communication with humans is crucial. Empathy plays a fundamental role in facilitating meaningful connections between individuals and robots, especially in situations where emotional support is essential for individuals' psychological well-being, such as in the case of *Autism Spectrum Disorders* (ASD). The cognitive theory of Mindblindness (Baron-Cohen, 1997), which focuses on the social and communicative difficulties associated with autism, states that individuals with ASD demonstrate limited ability to recognize emotions and mental states, hindering their social interaction [3]. Robots, perceived as non-human interlocutors but endowed with a certain human likeness, can play a significant role in helping autistic individuals develop and practice social and emotional skills in a safe and controlled environment. The quality of empathetic interaction depends directly on the precision and accuracy with which the robot controls its expressions. The greater the degree of control exerted, the more realistic and engaging the robot's behavior will be for humans, positively influencing the effectiveness of human-robot interactions, especially in sensitive areas such as autism management. The research conducted by De Rossi et al., further advanced the therapeutic approach based on the theory of mindblindness, focusing on the application of social robots in autism therapy [19]. The use of social robots offers a unique opportunity to provide personalized and interactive support for individuals with autism spectrum disorders, helping to overcome challenges related to the recognition of emotions and mental states. Through the simulation of social scenarios and interaction with the robot, patients can develop and practice social skills in a controlled and comfortable environment. This method not only enhances the social skills of individuals with autism but can also promote their emotional well-being and social integration. In this work, we propose a novel approach to the problem of the inverse kinematic for a humanoid robotic face using a deep learning framework, detailed in Section II, where a facial-recognition pre-trained model is fine-tuned for the specific task. We developed the framework for an advanced humanoid robot, named Abel (Cominelli et al., [6]), whose details are explained in Section III, with an incredible capacity of reproducing emotional facial expression, due to the high-degrees of freedom of its face. Finally, in Section IV, experimental results of the proposed framework are described.

II. Proposed work

A. Overall architecture

The presented framework incorporates a cascading use of two deep neural network models: *Multi-task Cascaded Convolutional Networks* (MTCNN) and InceptionResnetV1. MTCNN is a network designed for accurate face detection and alignment, leveraging a cascaded structure with three stages of deep convolutional networks to detect facial landmarks and faces [25]. Following MTCNN, InceptionResnetV1, also known as FaceNet, is employed for facial recognition. It combines the Inception architecture with residual connections to enhance learning, and is trained to generate embeddings of faces that capture the facial features in a high-dimensional space, facilitating accurate identification and verification of individuals [22]. In the proposed framework, MTCNN is utilized to crop a specific section around the robot's face with a set dimension, serving as a preparatory step for feature extraction. Subsequently, InceptionResnetV1 has been adapted to accept these cropped images from MTCNN as inputs, and finetuned to output values corresponding to the servo-motor configurations of the Abel's face. Both module are provided in `facenet-pytorch`, a PyTorch library widely used in computer vision and security applications, including access control systems, surveillance, and social media for automatic photo tagging. The overall pipeline of the proposed framework is shown in Fig. 1.

B. CNN

MTCNN and InceptionResnetV1 modules exploit a *Convolutional Neural Network* (CNN) architecture, designed for efficiently handling data with a grid-like structure, such as 1D time series or 2D image pixels [8]. The foundational work of LeCun et al., particularly with the LeNet network for recognizing handwritten digits, introduced the use of the backpropagation algorithm in CNNs [13]. This laid the groundwork for subsequent advancements, notably the development of AlexNet, the pioneering deep convolutional neural network that significantly advanced image classification [11]. A CNN architecture is structured with several layers, each with a distinct role in processing and deriving significant features from the input. Convolutional layers, equipped with multiple filters or kernels, traverse the input data to perform convolutions, identifying specific patterns and spatial information, thereby producing feature maps. To introduce non-linearity, an activation function such as *Rectified Linear Unit* (ReLU) or its variants like Leaky ReLU or Parametric ReLU is applied post-convolution. Following convolutional layers, pooling layers serve to diminish the dimensionality and spatial size of the feature maps, streamlining the complexity by summarizing critical information. Typically, max pooling selects the maximum value within a specific window of the features, as the representative for that area. The network then flattens the feature maps into a one-dimensional vector,

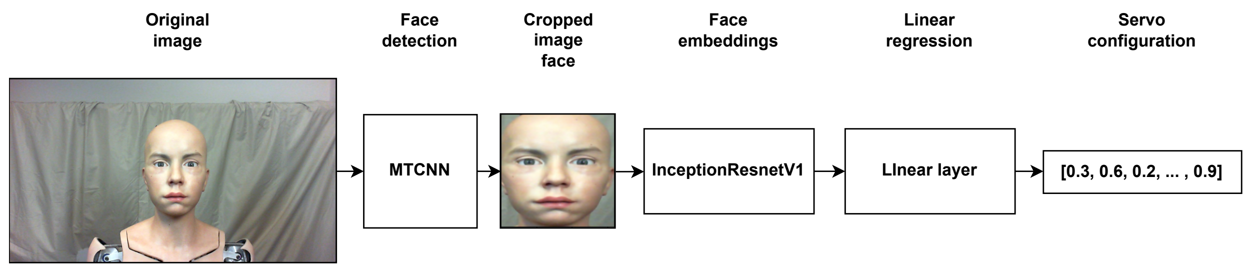


Fig. 1: Pipeline of the proposed framework.

which proceeds to *Fully Connected (dense) Layers* (FCLs). These layers are tasked with making predictions or classifications from the features, utilizing activation functions to maintain non-linearity. The CNN concludes with an output layer, formulating the predictions with an activation function suited to the task, such as Softmax for classification, which computes a probability distribution across various classes [14]. A loss function evaluates the difference between the network predictions and the actual labels, with the choice of loss function tailored to the task, like categorical cross-entropy for multi-class classification [26]. The training phase involves fine-tuning the parameters of the network to minimize the loss through optimization algorithms such as *stochastic gradient descent* (SGD) or Adam optimizer, enhancing the network accuracy through backpropagation [10]. Despite the complexity of CNN structures, illustrated in Fig. 2, contemporary machine learning platforms like PyTorch facilitate the implementation of the CNN operations.

C. MTCNN

The MTCNN model, as implemented in the `facenet-pytorch` library, is designed for efficient face detection and recognition. It comprises three sequential stages:

1) **P-Net (Proposal Network)**: Generates candidate windows for faces across multiple scales, outputting bounding boxes and confidence scores;
2) **R-Net (Refine Network)**: Refines the bounding boxes from the P-Net, eliminating many false positives and providing updated confidence scores;
3) **O-Net (Output Network)**: Performs further refinement on bounding boxes and outputs facial landmarks alongside confidence scores.

This cascaded architecture allows for a fine-grained approach to face detection, making the `facenet-pytorch` implementation of MTCNN a popular choice in various applications. In the proposed image processing pipeline, the MTCNN model is employed, as implemented in the `facenet-pytorch` library, to detect and crop the face of Abel from input images. This approach allows us to focus exclusively on the regions of interest, thereby enhancing the efficiency of subsequent facial analysis and recognition phases. The MTCNN model is configured with specific parameters to optimize face detection according to the characteristics of our dataset and the requirements of our pipeline. The used parameters are as follows:

- **image_size**: The size of the cropped face images. Set to 160 pixels, ensuring a standard dimension for all processed faces, which aids in maintaining consistency in the recognition phase;
- **margin**: The margin around the detected face. We opted for a 0 margin, meaning the cropping is tight around the detected face boundaries;
- **min_face_size**: The minimum size of faces to be detected. Set to 20 pixels to ensure that even smaller faces within the images are detected and processed;
- **thresholds**: The detection thresholds for the three stages of the MTCNN. Set to [0.6, 0.7, 0.7], these thresholds balance the trade-off between detection accuracy and the number of false positives;
- **factor**: The scale factor for the image pyramid used in face detection. Set to 0.709, it controls the scale at which the image is rescaled at each step, influencing the detection of faces of various sizes;
- **post_process**: Indicates whether to apply a post-processing step to the cropped faces. Set to True, ensuring that the cropped images are properly aligned and enhanced for better recognition results.

Table I shows the proper configuration of the MTCNN parameters used in the proposed work pipeline, crucial in tailoring the face detection process to our specific needs, ensuring high-quality cropping of Abel's face for further analysis within our framework pipeline.

TABLE I: MTCNN parameters used in the proposed work pipeline

Parameter	**Value**
`image_size`	160
`margin`	0
`min_face_size`	20
`thresholds`	[0.6, 0.7, 0.7]
`factor`	0.709
`post_process`	True

D. InceptionResnetV1

In the proposed image processing pipeline, we leverage the InceptionResNetV1 model, a renowned architecture known for its high performance in facial recognition tasks. This model is part of the `facenet-pytorch` library and is instrumental in processing images cropped by the MTCNN model. Upon receiving cropped face images, the InceptionResNetV1 model

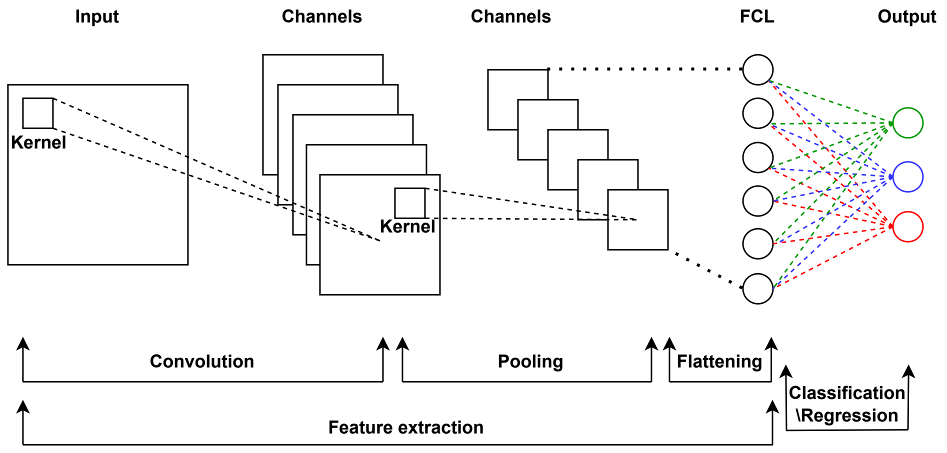


Fig. 2: Schematic structure of CNN.

transforms these images into a high-dimensional embedding. This embedding effectively captures the unique features of the face, serving as a distinctive representation that can be used for various facial analysis tasks. The model employed in our pipeline is pretrained on the `vggface2` dataset, offering a robust foundation for facial feature extraction. To adapt the model for our specific application, namely generating configurations for servos based on facial features, we introduced a modification to the model output layer. Specifically, we appended a linear layer with 22 outputs to the existing architecture. This layer is designed to map the high-dimensional face embeddings to a 22-dimensional space, corresponding to the configurations of the servos. To constrain the servo configuration values between 0 and 1, ensuring they remain within a valid range, the output of the linear layer is passed through a sigmoid activation block. This operation transforms the linear layer outputs into a set of values that are interpretable as servo configurations, enabling precise control over the servos based on the detected facial features. The pretrained InceptionResNetV1 model was fine-tuned using images of Abel's face, tailoring the model to our robot's specific facial characteristics. This fine-tuning process ensures the resulting model embeddings to be highly relevant and accurate for the mapping task. The *Mean Absolute Error Loss* ($MAELoss$) was employed as loss function. This loss is well-suited for regression tasks like the one proposed in this work, where the goal is to predict servo configurations as accurately as possible. Given $\boldsymbol{\theta}_i$, the target servo position vector of the i-th sample, and $\hat{\boldsymbol{\theta}}^k_{m,i}$, the predicted servo position vector by the InceptionResnetV1 training model m at time step k, the $MSELoss$ between these two vectors is defined as:

$$MSELoss(\boldsymbol{\theta}_i, \hat{\boldsymbol{\theta}}^k_{m,i}) \triangleq \frac{1}{n_s}\sum_{s=1}^{n_s}(\theta_{i,s} - \hat{\theta}^k_{m,i,s})^2 \tag{1}$$

where:

- n_s is the dimension of the vectors $\boldsymbol{\theta}_i$ and $\hat{\boldsymbol{\theta}}^k_{m,i}$, indicating the number of components (or servos) in the servo position vector;
- $\theta_{i,s}$ represents the s-th component of the actual servo position vector $\boldsymbol{\theta}_i$ for the i-th sample;
- $\hat{\theta}^k_{m,i,s}$ is the s-th component of the predicted vector $\hat{\boldsymbol{\theta}}^k_{m,i}$ by model m at time step k.

This version of the $MSELoss$ specifically evaluates the accuracy of the predictions made by model m for the i-th sample at time step k, by calculating the mean of the squares of the differences between each component of the target servo position vector and its corresponding predicted component. Lower values of this loss function indicate that model m is able to predict the servo positions more accurately at time step k, while higher values indicate less accurate predictions. The employed optimizer is Adam, for its effectiveness in handling sparse gradients and its adaptive learning rate capabilities. The fine-tuning training of the proposed InceptionResnetV1 model was not a one-off process; it involved multiple iterations with varying hyperparameters to optimize performance. The intricacies of the hyperparameter tuning process, including the range of values explored for each parameter and the methodology employed for evaluating model performance, is presented in Subsection III-C. Through iterative training and hyperparameter tuning, we developed a robust system capable of translating facial features into precise servo configurations.

III. Experimental setup

A. Abel

The design of Abel's body was strongly influenced by the importance of being emotionally expressive, a concept that has guided the entire process. This emphasis on emotional expressiveness has been developed based on years of experience with the *Facial Automaton for Conveying Emotions* (FACE) robot, a social robot with human-inspired facial expressiveness that is used in therapy with children with autism spectrum disorder and in educational contexts as a synthetic tutor [20]. Abel's face plays a crucial role in interaction with humans, thanks to motors that allow precise control of facial expressions. Its expressiveness not only gives the robot a more human-like appearance but also optimizes interaction with humans, making it more natural and engaging. Collaboration between engineers and creatives with such artistic inspiration has been crucial in this regard, as mechatronics is dictated by the shape of the robot's face and body, as well as its conceptual design. The $n_s = 22$ facial motors, housed in Abel's skull, allow a wide range of realistic and refined facial movements, from executing glances to simulating speech through a system of mechanical movement transmission and control module. Specifically, 4 move the forehead, 1 for the jaw, 4 for the mouth, 2 for the cheeks, 1 for the chin (Futaba BLS 173 SV), 8 for the eyes, and 2 for the lips (MKS HV 93 and MKS HV6130). Additionally, 5 motors are dedicated to the neck and head movement (Dynamixel XM 540-W270-T). Finally, in each arm, 3 motors are mounted for the shoulder, 1 for the elbow, 1 for rotating the arm (Dynamixel XM 540-W270-T), and 3 motors in each hand (MKS HV6130H), for a total of 43 degrees of freedom (for more details on motors, see Table II and Table III, for a graphic representation of the motors of the face, see Fig. 3). Futaba BLS motors offer a combination of high speed, torque, and precision, making them suitable for applications requiring rapid and precise movements. Their reliability and durability make them a valid choice for a robot like Abel. MKS motors are known for their robustness and power, making them ideal for movements requiring greater strength and resistance. Their ability to handle heavy loads makes them suitable for applications requiring more vigorous and detailed movements. Control of Abel's facial motors is entrusted to two Pololu Mini Maestro 12-channel controllers. This device allows synchronized and precise management of movements, necessary for precise and smooth control of position, speed, and acceleration of connected servo motors and consequently facial expressions. It also has a USB interface for programming and control of the module through dedicated software and allows the use of communication protocols such as TTL serial, I²C, and USB to integrate the module with other devices and microcontrollers. In particular, this control module is chosen for its ability to simultaneously manage up to 12 output channels and for advanced programming options, features that make it ideal for integration with Abel's control system. In summary, the combination of high-quality motors and a sophisticated control system ensures that Abel is capable of expressing a wide range of emotions realistically and engagingly in its daily interactions.

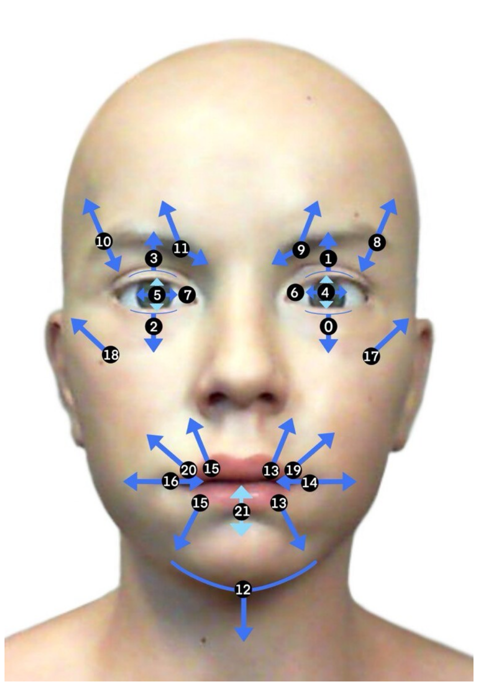


Fig. 3: Arrangement of servomotors for face control.

TABLE II: Facial expression servomotors

ID motor	Function	Servo model
0	Eyelid lower Lx	MKS HV6130
1	Eyelid top Lx	MKS HV6130
2	Eyelid lower Rx	MKS HV6130
3	Eyelid top Rx	MKS HV6130
4	Eye direction U/D Lx	MKS HV 93
5	Eye direction U/D Rx	MKS HV 93
6	Eye direction I/O Lx	MKS HV 93
7	Eye direction I/O Rx	MKS HV 93
8	Outer brow Lx	Fut. BLS 173 SV
9	Inner brow Lx	Fut. BLS 173 SV
10	Outer brow Rx	Fut. BLS 173 SV
11	Inner brow Rx	Fut. BLS 173 SV
12	Jaw	Fut. BLS 172 SV
13	Mouth corner U/D Lx	Fut. BLS 173 SV
14	Mouth corner I/O Lx	Fut. BLS 173 SV
15	Mouth corner U/D Rx	Fut. BLS 173 SV
16	Mouth corner I/O Rx	Fut. BLS 173 SV
17	Cheek Lx	Fut. BLS 173 SV
18	Cheek Rx	Fut. BLS 173 SV
19	Lip top Lx	MKS HV 93
20	Lip top Rx	MKS HV 93
21	Chin	Fut. BLS 173 SV

B. Dataset

The dataset of robot faces was created following a systematic approach to capture the variability in robot expressions. To assemble this dataset, a Microsoft LifeCam HD-6000 camera was used, positioned at a fixed distance from the robot, in a controlled light disposition setting to avoid shading issues. A total of $n_i = 231$ images were captured. These images represent various random configurations of the robot servo position vector $\boldsymbol{\theta}$, each one drawn from an uniform probability distribution function. Each element of the position vector $\boldsymbol{\theta}$ assume value between 0 and 1, corresponding to the normalized *Pulse Width Modulation* (PWM) signal values, representing the minimum and maximum achievable angles for the servos. This method ensures a consistent and controlled variation in the robot facial expressions, providing a rich dataset for the proposed application. Thus, the dataset consists of a tuple $\mathcal{D} = \{\boldsymbol{X}_i, \boldsymbol{\theta}_i\}_{i=1}^{n_i}$, where $\boldsymbol{X}_i \in \mathbb{R}^{n_w \times n_h}$ with height $n_h = 480$ pixel and width $n_w = 640$ pixel, is the i-th image and $\boldsymbol{\theta}_i \in \mathbb{R}^{n_s}$ is the i-th servo configuration vector. A sample of cropped Abel's faces from the original dataset is shown in Fig. 4a. For the training of the modified InceptionResnetV1 (proposed in Subsection II-D), we split the dataset in 80% for the training set, 10% for the validation set and 10% for the test set.

TABLE III: Body servomotors

ID motor	Function	Servo model
0	Shoulder joint frontal Lx	Dyn. XM 540-W270-T
1	Shoulder joint lateral Lx	Dyn. XM 540-W270-T
2	Upper arm twist Lx	Dyn. XH 430-W350-T
3	Arm elbow Lx	Dyn. XM 540-W270-T
4	Lower arm twist Lx	Dyn. XL 430-W250-T
5	Shoulder joint frontal Rx	Dyn. XM 540-W270-T
6	Shoulder joint lateral Rx	Dyn. XM 540-W270-T
7	Upper arm twist Rx	Dyn. XH 430-W350-T
8	Arm elbow Rx	Dyn. XM 540-W270-T
9	Lower arm twist Rx	Dyn. XL 430-W250-T
10	Wrist Lx	MKS HV6130H
11 - 12	Fingers ($\times 2$) Lx	MKS HV6130H
13	Wrist Rx	MKS HV6130H
14 - 15	Fingers ($\times 2$) Rx	MKS HV6130H
16	Head twist	Dyn. XM 540-W270-T
17	Upper neck Lx	Dyn. XH 430-W350-T
18	Upper neck Rx	Dyn. XH 430-W350-T
19	Lower neck Lx	Dyn. XM 540-W270-T
20	Lower neck Rx	Dyn. XM 540-W270-T

C. Hyperparameter optimization

Before training our machine learning model, we undertook a comprehensive hyperparameter optimization process. This was facilitated by the utilization of the *Weights & Biases* (wandb) tool, allowing us to systematically explore a range of values for key parameters and ascertain their optimal settings. Particularly for data augmentation-related parameters, our approach involved varying each parameter within a range that extends from 0 to the specified maximum value. This strategy ensured that each iteration of the training process incorporated data augmentation to a degree dictated by the maximum value, thereby enhancing the robustness and generalization capability of our model. The following list details the parameters that subject to be variated during the optimization process:

- **learning_rate**: Explored values included 0.001, 0.0001, and 0.00001, adjusting the pace at which the model learns;
- **batch_size**: Considered values were 8, 16, and 32, determining the number of training samples to process before updating the model's internal parameters;
- **max_epochs**: The values set for exploration were 10, 50, 100, and 300, establishing the maximum iterations over the complete training dataset;
- **apply_transform_probability**: This parameter was adjusted within 0, 0.5, 0.75, and 1.0, governing the probability of applying transformations such as rotations and translations to the input data during training;
- **affine_degrees**: For image rotation in data augmentation, the range was set between 0 and the maximum values of 10, 20, and 30 degrees;
- **affine_translate**: The translation of images in data augmentation was confined within a range from 0 to maximum values of 0.1, 0.2, and 0.3, expressed as a fraction of image dimensions;
- **affine_scale**: Scaling of images in data augmentation was varied from 0 up to maximum values of 0.1, 0.2, and 0.3;
- **color_brightness**: The brightness adjustment in color augmentation of images was explored from 0 up to maximum values of 0.1, 0.2, and 0.3;
- **color_contrast**: For altering the contrast in color augmentation of images, the range was from 0 to maximum values of 0.1, 0.2, and 0.3;
- **color_saturation**: The saturation in color augmentation of images was varied from 0 up to maximum values of 0.1, 0.2, and 0.3;
- **color_hue**: The hue adjustment in color augmentation of images was set within a range from 0 to maximum values of 0.1, 0.2, and 0.3;

These parameters and their respective values, which were explored during hyperparameter optimization, are summarized in Table IV. Examples of generated augmented data from the original dataset are shown in Fig. 4b.

TABLE IV: Hyperparameter optimization values

Parameter	Values
`learning_rate`	0.001, 0.0001, 0.00001
`batch_size`	8, 16, 32
`max_epochs`	10, 50, 100, 300
`apply_transforms_probability`	0.0, 0.5, 0.75, 1.0
`affine_degrees`	0, 10, 20, 30
`affine_translate`	0.0, 0.1, 0.2, 0.3
`affine_scale`	0.0, 0.1, 0.2, 0.3
`color_brightness`	0.0, 0.1, 0.2, 0.3
`color_contrast`	0.0, 0.1, 0.2, 0.3
`color_saturation`	0.0, 0.1, 0.2, 0.3
`color_hue`	0.0, 0.1, 0.2, 0.3

D. Metrics and data evaluation methods

Suppose the following notations:

- n_i represents the number of samples or data points in the dataset;
- n_s represents the number of servos;
- $\boldsymbol{\theta}_i$ refers to the target servo position vector of the i-th data sample;
- $\hat{\boldsymbol{\theta}}_{m,i}$ refers to the predicted servo position vector from a trained model m, given the image $\boldsymbol{X}_i$ as input;
- $\theta_{i,s}$ refers to the s-th target servo position related to the i-th target servo position vector;
- $\hat{\theta}_{m,i,s}$ refers to the s-th predicted servo position related to the i-th predicted servo position vector from the trained model m;
- $\boldsymbol{\Theta}$ is the matrix of size $n_i \times n_s$ whose raws are the vectors $\boldsymbol{\theta}_i$;
- $\hat{\boldsymbol{\Theta}}_m$ is the matrix of size $n_i \times n_s$ whose raws are the vectors $\hat{\boldsymbol{\theta}}_{m,i}$;
- $\boldsymbol{\Theta}_s$ is the s-th column of $\boldsymbol{\Theta}$;
- $\hat{\boldsymbol{\Theta}}_{m,s}$ is the s-th column of $\hat{\boldsymbol{\Theta}}_m$;
- $\Theta_{s,i}$ is the i-th element of $\boldsymbol{\Theta}_s$;
- $\hat{\Theta}_{m,s,i}$ is the i-th element of $\hat{\boldsymbol{\Theta}}_{m,s}$.

then, the following metrics are used as benchmarks for the hyperparameter optimization:

Mean Squared Error (MSE): measures the average squared difference between the predicted and the target values:

$$MSE(\boldsymbol{\theta}_i, \hat{\boldsymbol{\theta}}_{m,i}) \triangleq \frac{1}{n_s} \sum_{s=1}^{n_s} (\theta_{i,s} - \hat{\theta}_{m,i,s})^2 \tag{2}$$

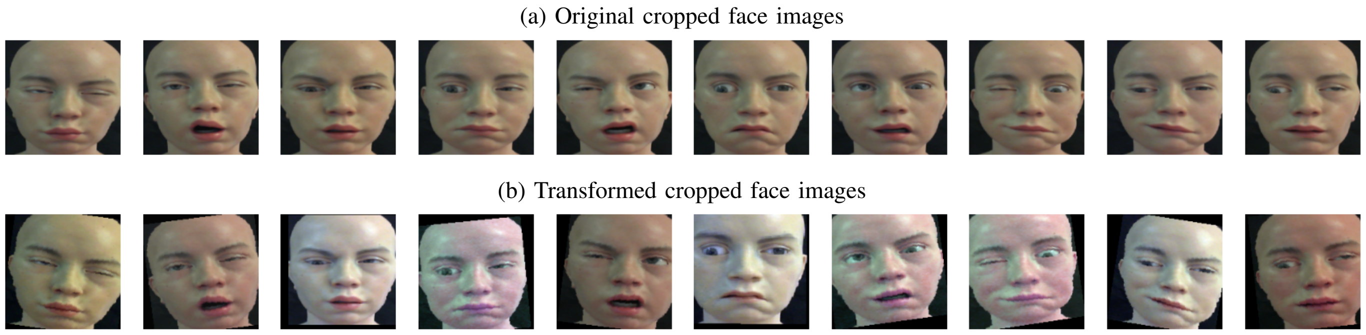

(a) Original cropped face images
(b) Transformed cropped face images

Fig. 4: Subset of data samples of cropped images of Abel's face (a) paired with an example of related augmentations (b).

Average Sample - Mean Squared Error (AS-MSE): calculates the MSE for each servo across all samples:

$$AS\text{-}MSE(\mathbf{\Theta}_s, \hat{\mathbf{\Theta}}_{m,s}) \triangleq \frac{1}{n_i}\sum_{i=1}^{n_i}(\Theta_{s,i} - \hat{\Theta}_{m,s,i})^2 \tag{3}$$

Average Overall - Mean Squared Error (AO-MSE): combines the MSE across all servos and samples:

$$\begin{aligned} AO\text{-}MSE(\boldsymbol{\theta}_i, \hat{\boldsymbol{\theta}}_{m,i}) &\triangleq \frac{1}{n_i}\sum_{i=1}^{n_i} MSE(\boldsymbol{\theta}_i, \hat{\boldsymbol{\theta}}_{m,i}) \\ &= \frac{1}{n_s}\sum_{s=1}^{n_s} AS\text{-}MSE(\mathbf{\Theta}_s, \hat{\mathbf{\Theta}}_{m,s}) \end{aligned} \tag{4}$$

Root Mean Squared Error (RMSE): is the square root of the average squared differences between the predicted and target values:

$$RMSE(\boldsymbol{\theta}_i, \hat{\boldsymbol{\theta}}_{m,i}) \triangleq \sqrt{\frac{1}{n_s}\sum_{s=1}^{n_s}(\theta_{i,s} - \hat{\theta}_{m,i,s})^2} \tag{5}$$

Average Sample - Root Mean Squared Error (AS-RMSE): calculates the RMSE for each servo across all samples

$$AS\text{-}RMSE(\mathbf{\Theta}_s, \hat{\mathbf{\Theta}}_{m,s}) \triangleq \sqrt{\frac{1}{n_i}\sum_{i=1}^{n_i}(\Theta_{s,i} - \hat{\Theta}_{m,s,i})^2} \tag{6}$$

Average Overall - Root Mean Squared Error (AO-RMSE): combines the RMSE across all servos and samples:

$$\begin{aligned} AO\text{-}RMSE(\boldsymbol{\theta}_i, \hat{\boldsymbol{\theta}}_{m,i}) &\triangleq \frac{1}{n_i}\sum_{i=1}^{n_i} RMSE(\boldsymbol{\theta}_i, \hat{\boldsymbol{\theta}}_{m,i}) \\ &= \frac{1}{n_s}\sum_{s=1}^{n_s} AS\text{-}RMSE(\mathbf{\Theta}_s, \hat{\mathbf{\Theta}}_{m,s}) \end{aligned} \tag{7}$$

Mean Absolute Error (MAE): measures the average magnitude of errors in a set of predictions:

$$MAE(\boldsymbol{\theta}_i, \hat{\boldsymbol{\theta}}_{m,i}) \triangleq \frac{1}{n_s}\sum_{s=1}^{n_s}|\theta_{i,s} - \hat{\theta}_{m,i,s}| \tag{8}$$

Average Sample - Mean Absolute Error (AS-MAE): calculates the MAE for each servo across all samples:

$$AS\text{-}MAE(\mathbf{\Theta}_s, \hat{\mathbf{\Theta}}_{m,s}) \triangleq \frac{1}{n_i}\sum_{s=1}^{n_s}|\Theta_{s,i} - \hat{\Theta}_{m,s,i}| \tag{9}$$

Average Overall - Mean Absolute Error (AO-MAE): combines the MAE across all servos and samples:

$$\begin{aligned} AO\text{-}MAE(\boldsymbol{\theta}_i, \hat{\boldsymbol{\theta}}_{m,i}) &\triangleq \frac{1}{n_i}\sum_{i=1}^{n_i} MAE(\boldsymbol{\theta}_i, \hat{\boldsymbol{\theta}}_{m,i}) \\ &= \frac{1}{n_s}\sum_{s=1}^{n_s} AS\text{-}MAE(\mathbf{\Theta}_s, \hat{\mathbf{\Theta}}_{m,s}) \end{aligned} \tag{10}$$

R-squared (R²): indicates the proportion of the variation in a set of predictions:

$$R^2(\boldsymbol{\theta}_i, \hat{\boldsymbol{\theta}}_{m,i}) \triangleq 1 - \frac{\sum_{s=1}^{n_s}(\theta_{i,s} - \hat{\theta}_{m,i,s})^2}{\sum_{s=1}^{n_s}(\theta_{i,s} - \bar{\theta}_{m,i})^2} \tag{11}$$

where $\bar{\theta}_{m,i}$ is the average of predicted servo positions across all servos, expressed as:

$$\bar{\theta}_{m,i} \triangleq \frac{1}{n_s}\sum_{s=1}^{n_s}\hat{\theta}_{m,i,s} \tag{12}$$

Average Sample - R-squared (AS-R²): calculates the R² for each servo across all samples:

$$AS\text{-}R^2(\mathbf{\Theta}_s, \hat{\mathbf{\Theta}}_{m,s}) \triangleq 1 - \frac{\sum_{i=1}^{n_i}(\Theta_{s,i} - \hat{\Theta}_{m,s,i})^2}{\sum_{i=1}^{n_i}(\Theta_{s,i} - \bar{\Theta}_{m,s})^2} \tag{13}$$

where $\bar{\Theta}_{m,s}$ is the average of predicted servo positions across all samples, expressed as:

$$\bar{\Theta}_{m,s} \triangleq \frac{1}{n_i}\sum_{i=1}^{n_i}\hat{\Theta}_{m,s,i} \tag{14}$$

Average Overall - R-squared (AO-R²): combines the R² across all servos and samples:

$$\begin{aligned} AO\text{-}R^2(\boldsymbol{\theta}_i, \hat{\boldsymbol{\theta}}_{m,i}) &\triangleq \frac{1}{n_i}\sum_{i=1}^{n_i} R^2(\boldsymbol{\theta}_i, \hat{\boldsymbol{\theta}}_{m,i}) \\ &= \frac{1}{n_s}\sum_{s=1}^{n_s} AS\text{-}R^2(\mathbf{\Theta}_s, \hat{\mathbf{\Theta}}_{m,s}) \end{aligned} \tag{15}$$

In the next section (Section IV), to analyze the data of hyperparameter optimization, comparisons between target and predicted servo positions are plotted, as well as estimated *Probability Density Functions* (PDFs) using the *Kernel Density Estimation* (KDE) method. KDE is a non-parametric method used to estimate the PDF of a random variable. For a generic random variable x the estimated PDF $\hat{f}(x)$ expressed by (16), calculated with the `sns.kdeplot()` method of *Seaborn*, a python library, is given by the sum of kernel functions centered on each data point, normalized so that its integral over the entire space equals 1. Mathematically, the formula is:

$$f(x) = \frac{1}{n\omega} \sum_{i=1}^{n} K\left(\frac{x - x_i}{\omega}\right) \tag{16}$$

where:

- n is the number of observations (data points),
- x_i are the data points,
- ω is the bandwidth that controls the smoothness of the PDF curve,
- K is the kernel function, which measures the distance between the observation point x and each data point x_i in terms of probability. In the case of Seaborn, the Gaussian kernel is often used, which has the form:

$$K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2} \tag{17}$$

The selection of the kernel K and the bandwidth ω is pivotal in the KDE estimation. Seaborn automates the choice of ω with preset values based on widely accepted heuristics, but users have the option to modify ω to affect the resulting PDF's smoothness through the `bw_adjust` parameter in `sns.kdeplot()`. In Section IV, we compare the estimated PDFs of the random variables $\bar{\theta}$ with $\bar{\theta}_m$, whose observed data are $\bar{\theta}_i$ and $\bar{\theta}_{m,i}$, respectively, where:

$$\bar{\theta}_i \triangleq \frac{1}{n_s} \sum_{s=1}^{n_s} \theta_{i,s} \tag{18}$$

We also compare estimated PDF of the $AO\text{-}MAE$ using the KDE method applied to data resulting from a Monte Carlo evaluation. Experimental results related to the single servos performance evaluations are provided in the supplementary materials of this article.

IV. Experimental results

A. Hyperparameter optimization

We ran $N = 500$ simulations with different values of the hyperparameters. All simulations were run on a cluster of 4 NVIDIA A100 *graphics process units* (GPUs). Consider m_r to be the r-th model related to the r-th simulation and to evaluate the best model through the best performance on the validation set. The model $m = m^*$ with the minimum value of $AO\text{-}MAE$ for the validation set is m_{408}. The setting of hyperparameters related to m^* is shown in Table V. We can see how the parameter `affine_transforms_probability` is 0, meaning that the optimization process excludes the augmentation for the model. The overall model evaluation for the hyperparameter optimization, is shown in Fig. 5, where the $AO\text{-}MAE$ for the validation set of each model m_r is expressed over the *days of execution* (d). The value related to the model m^* is highlighted with a red dot. The related performance metrics of m^* are listed in Table VI. Results demonstrate how MAE metrics are lower than 0.10, meaning that the model m^* is able to predict the Abel's face servo positions with an overall error lower than 10%. $MSELoss$ comparison over the iterations for the training and validation set, during the training of m^*, is shown in Fig. 6, where the i_T denotes that the sample $\boldsymbol{\theta_{i_T}}$ belongs to the training set, and the V in i_V denotes that the sample $\boldsymbol{\theta_{i_V}}$ belongs to the validation set.

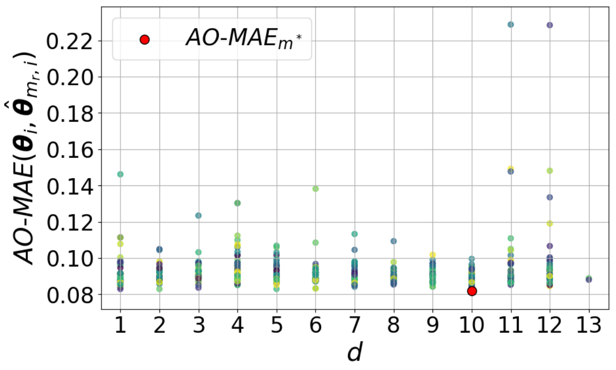


Fig. 5: Overall evaluation for the hyperparameter optimization of the proposed fine-tuned InceptionResnetV1 model.

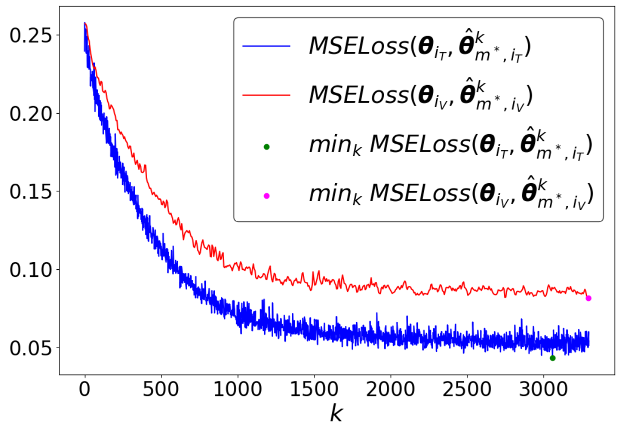


Fig. 6: Comparison of $MAELoss$ over k for the validation and training set, during the training of m^* during the hyper-parameter optimization process.

B. Monte Carlo evaluation

Despite the hyperparameter optimization excluded the augmentation data process, the stochasticity of the model during the training phase still persists due to the randomicity of the initial weight distribution and the shuffle during the batch splitting process. Thus, for a robust performance evaluation of the hyperparameter configuration related to the model m^* we conducted a Monte Carlo evaluation of $M = 180$ multiple model training runs with the same hyperparameter values of

TABLE V: Hyperparameter configuration of m^*

Parameter	Value
affine_degrees	20
affine_scale	0.1
affine_translate	0
apply_transforms_prob	0
batch_size	32
color_brightness	0.3
color_contrast	0
color_hue	0.1
color_saturation	0.3
learning_rate	0.001
max_epochs	300

TABLE VI: Overall performance metrics of m^* for the run of the hyperparameter optimization.

dataset	$AO\text{-}MSE_{m^*}$	$AO\text{-}RMSE_{m^*}$	$AO\text{-}MAE_{m^*}$	$AO\text{-}R^2_{m^*}$
train	0.001	0.036	0.028	0.984
val.	0.015	0.123	0.081	0.823
test	0.008	0.092	0.067	0.898

the model m^* (listed in Table V). For each model $m = m^\circ_r$, where r is the index of the r-th trial, we evaluated the average metrics related to $m = m^\circ$, the generic model with the same hyperparameter configuration of m^*. Indeed, the symbol $\circ$ denotes the metrics are averaged over the set of all the M Monte Carlo runs. From the resulting data, the plots of the PDFs of the $AO\text{-}MAE_{m^\circ}$ for the train, validation and test set, are shown in Fig. 7. In this figures, notation A, denotes the area underlying the curve of the estimated PDFs. Notations TR (train), TE (test) e V (validation) denote the type of dataset over which the metrics are calculated. The average of the MAE distribution, along with all the other metrics are shown in Table VI. Fig. 8 show the distribution of the target values versus the predicted values. Finally, Fig. 9 show the estimated PDFs of the target versus the predicted values. Results prove how the hyperparameter configuration of m^* is able to train models that perform with an average overall MAE error lower than 10% for both the validation and test set. Finally, the output of an instance (i.e., with the same hyperparameter configuration) of the model m°, given an image of the test set, is shown in Fig. 10.

V. Conclusion

This research work successfully developed a comprehensive framework for training inverse kinematics models capable of accurately mapping facial expressions to servo positions in Abel, a robot with a high degree of facial movement complexity. By leveraging a pre-trained InceptionResnetV1 model, which was further fine-tuned with images of Abel's face displaying various expressions linked to random servo position vectors, we established a robust method to obtain this. The optimal configuration of model hyperparameters was identified through a rigorous optimization process, and the effectiveness of this configuration was confirmed through statistical analysis of data resulting from a set of Monte Carlo simulations. The models demonstrated good performance both overall and in nearly all individual servo evaluations. This framework paves the way for the development of more advanced models that can enable robots to mimic and emulate human facial expressions, and simulate empathy, with a high level of detail. Such advancements hold promising implications for enhancing human-robot interaction, contributing significantly to the field of social robotics. In conclusion, our work not only showcases the potential of using deep learning techniques in robotics but also paves the way to research in creating more expressive and interactive robots.

TABLE VII: Overall evaluation metrics for the model m°, averaged over the Monte Carlo runs.

dataset	$AO\text{-}MSE_{m^\circ}$	$AO\text{-}RMSE_{m^\circ}$	$AO\text{-}MAE_{m^\circ}$	$AO\text{-}R^2_{m^\circ}$
train	0.001	0.037	0.029	0.982
val.	0.018	0.127	0.090	0.766
test	0.011	0.096	0.074	0.866

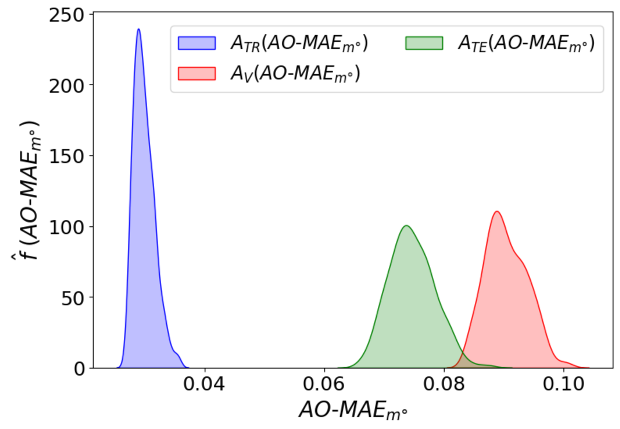


Fig. 7: Estimated PDFs of the $AO\text{-}MAE_{m^\circ}$ metric for the train, validation and test set.

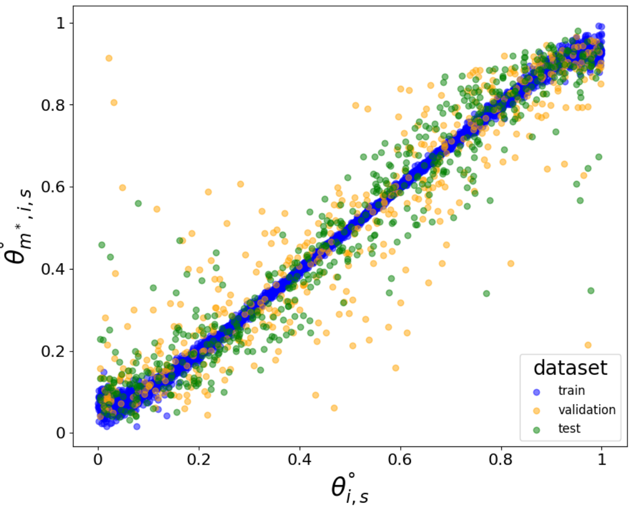


Fig. 8: Predicted versus target servo position values, averaged on the Monte Carlo runs.

Acknowledgment

The research leading to these results has received partial funding from the Italian Ministry of Education and Research

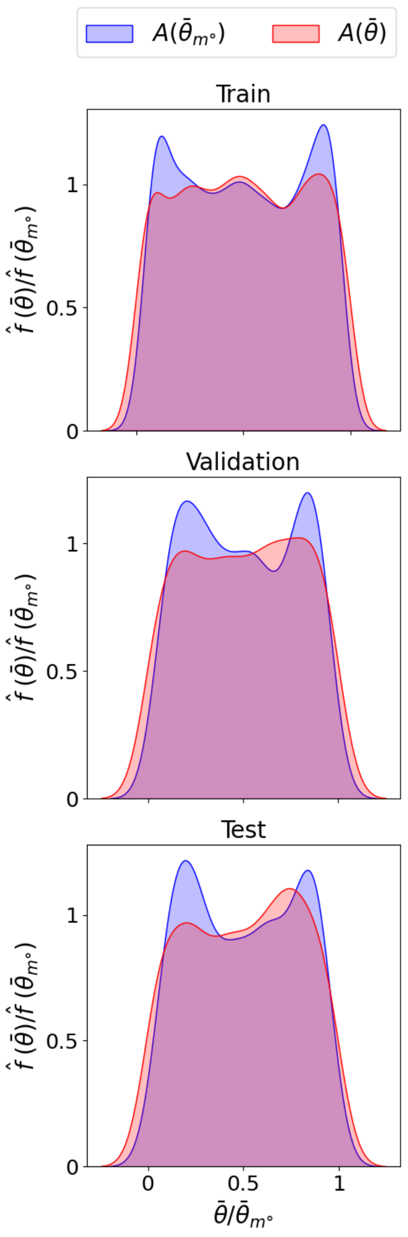


Fig. 9: Comparison of the estimated PDFs of the overall target servo positions of the train, validation and test set, versus the predicted servo positions, averaged over the Monte Carlo runs.

(MIUR) in the framework of the ForeLab project (Departments of Excellence), from PNRR - M4C2 - Investimento 1.3, Partenariato Esteso PE00000013- “FAIR- Future Artificial Intelligence Research” - Spoke 1 “Human-centered AI”, funded by the European Commission under the NextGeneration EU programme, and from the PRIN grant no. 2022ALBSWX of the Italian Ministry of University and Research.

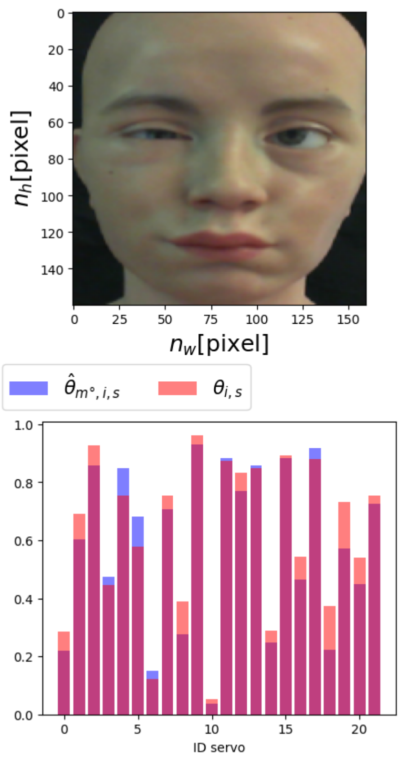


Fig. 10: Output of an instance of the generic model m°, given as input an image $\boldsymbol{X}_i$ of the test set.

References

[1] Francesco Aggogeri et al. “Inverse kinematic solver based on machine learning sequential procedure for robotic applications”. In: *Journal of Physics: Conference Series*. Vol. 2234. 1. IOP Publishing. 2022, p. 012007.

[2] Andreas Aristidou and Joan Lasenby. “Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver”. In: (2009).

[3] Simon Baron-Cohen. *Mindblindness: An essay on autism and theory of mind*. MIT press, 1997.

[4] Cynthia Breazeal. “Emotion and sociable humanoid robots”. In: *International Journal of Human-Computer Studies* 59.1 (2003). Applications of Affective Computing in Human-Computer Interaction, pp. 119–155. ISSN: 1071-5819. DOI: https://doi.org/10.1016/S1071-5819(03)00018-1. URL: https://www.sciencedirect.com/science/article/pii/S1071581903000181.

[5] Benjamin B Choi. *Inverse kinematics problem in robotics using neural networks*. Vol. 105869. Lewis Research Center, 1992.

[6] Lorenzo Cominelli, Gustav Hoegen, and Danilo De Rossi. “Abel: integrating humanoid body, emotions, and time perception to investigate social interaction and human cognition”. In: *Applied Sciences* 11.3 (2021), p. 1070.

[7] Bassam Daya, Shadi Khawandi, Mohamed Akoum, et al. “Applying neural network architecture for inverse

kinematics problem in robotics". In: *Journal of Software Engineering and Applications* 3.03 (2010), p. 230.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[9] Zhong Huang et al. "Facial Expression Imitation Method for Humanoid Robot Based on Smooth-Constraint Reversed Mechanical Model (SRMM)". In: *IEEE Transactions on Human-Machine Systems* 50.6 (2020), pp. 538–549. DOI: 10.1109/THMS.2020.3017781.

[10] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[12] Yasuaki Kuroe, Yasuhiro Nakai, and Takehiro Mori. "A new neural network approach to the inverse kinematics problem in robotics". In: *Proceedings 1993 Asia-Pacific Workshop on Advances in Motion Control*. IEEE. 1993, pp. 112–117.

[13] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[14] Weiyang Liu et al. "Large-margin softmax loss for convolutional neural networks". In: *Applied Bionics and Biomechanics, vol. 1* (2016).

[15] Jiaoyang Lu, Ting Zou, and Xianta Jiang. "A Neural Network Based Approach to Inverse Kinematics Problem for General Six-Axis Robots". In: *Sensors* 22.22 (2022), p. 8909.

[16] Emarc Magtanong et al. "Inverse kinematics solver for android faces with elastic skin". In: *Latest Advances in Robot Kinematics*. Springer. 2012, pp. 181–188.

[17] Aryslan Malik et al. "A Deep Reinforcement-Learning Approach for Inverse Kinematics Solution of a High Degree of Freedom Robotic Manipulator". In: *Robotics* 11.2 (2022). ISSN: 2218-6581. DOI: 10.3390/robotics11020044. URL: https://www.mdpi.com/2218-6581/11/2/44.

[18] Jeong Woo Park, Hui Sung Lee, and Myung Jin Chung. "Generation of Realistic Robot Facial Expressions for Human Robot Interaction". In: *Journal of Intelligent & Robotic Systems* 78.3 (June 2015), pp. 443–462. ISSN: 1573-0409. DOI: 10.1007/s10846-014-0066-1. URL: https://doi.org/10.1007/s10846-014-0066-1.

[19] Giovanni Pioggia et al. "FACE: Facial Automaton for Conveying Emotions". In: *Applied Bionics and Biomechanics* 1 (2004), p. 153078. ISSN: 1176-2322. DOI: 10.3233/ABB-2004-9693539. URL: https://doi.org/10.3233/ABB-2004-9693539.

[20] Giovanni Pioggia et al. "FACE: Facial automaton for conveying emotions". In: *Applied Bionics and Biomechanics* 1.2 (2004), pp. 91–100.

[21] Fuji Ren and Zhong Huang. "Automatic Facial Expression Learning Method Based on Humanoid Robot XIN-REN". In: *IEEE Transactions on Human-Machine Systems* 46.6 (2016), pp. 810–821. DOI: 10.1109/THMS.2016.2599495.

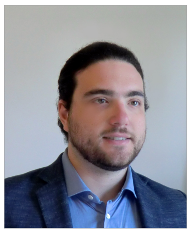
[22] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

[23] Bruno Siciliano, Oussama Khatib, and Torsten Kröger. *Springer handbook of robotics*. Vol. 200. Springer, 2008.

[24] Tingfan Wu et al. "Learning to Make Facial Expressions". In: *2009 IEEE 8th International Conference on Development and Learning, ICDL 2009* (Jan. 2009). DOI: 10.1109/DEVLRN.2009.5175536.

[25] Kaipeng Zhang et al. "Joint face detection and alignment using multitask cascaded convolutional networks". In: *IEEE signal processing letters* 23.10 (2016), pp. 1499–1503.

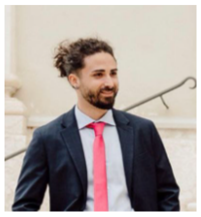
[26] Zhilu Zhang and Mert Sabuncu. "Generalized cross entropy loss for training deep neural networks with noisy labels". In: *Advances in neural information processing systems* 31 (2018).



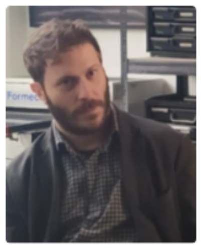
Graziano Alfredo Manduzio , received the B.Sc. degree in electronic and telecommunication engineering, the M.Sc. degree in automation engineering from the University of Florence, Florence, Italy, in 2013 and 2018 respectively. He received the Ph.D. degree in Smart Industry from the University of Pisa, Pisa, Italy, in 2023. In 2020, he was a Visiting Researcher with the NATO Science & Technology Organization (STO) Centre for Maritime Research and Experimentation (CMRE), La Spezia, Italy. He is currently a Postdoctoral Researcher at the Dipartimento di Ingegneria dell'Informazione (DII), University of Pisa and since 2022 he has been working in the team developing Abel, a humanoid robot with expressive emotional skills. His main research interests include social robotics, machine learning and Bayesian estimation.

Federico Galatolo , PhD in Information Engineering, is an Assistant Professor at the Dipartimento di Ingegneria dell'Informazione (DII) of the University of Pisa. His research has been cited by hundreds of peers and focuses on Artificial Intelligence, Bio-Inspired Architectures, Deep Learning, and Reinforcement Learning. He is an expert on modern neural architectures, convolutional neural networks, temporal neural networks, generative architectures, and semi-supervised architectures. He is co-founder of the "Machine Learning and Process Intelligence" research initiative. He is also an outspoken supporter of the Free Software movement and actively contributes to the development of different free software projects.

Mario Giovanni Cosimo Antonio Cimino , is an Associate Professor at the Dipartimento di Ingegneria dell'Informazione (DII) of the University of Pisa (Italy). His research lies in the areas of Information Systems and Artificial intelligence. He is (co-)author of more than 100 international scientific publications. He is an Associate Editor of the Journal of Granular Computing (Springer) and the Journal of Ambient Intelligence and Humanized Computing (Springer). He is Vice-Chair of the IEEE CIS Task Force "Intelligent Agents", IEEE Computational Intelligence Society. He was co-chair of the 2021 and 2022 IEEE Symposium on Intelligent Agents (IEEE SSCI).



Mattia Bruscia , earned his Bachelor's degree in Biomedical Engineering from the University of Pisa, where he is also pursuing a Master's degree in the same discipline. He is actively involved at the Enrico Piaggio Research Center, where he is supported by a scholarship funded by a PRIN award. Mattia's work with the humanoid robot "Abel" focuses on human-robot interaction. His research is dedicated to the application of machine learning and artificial intelligence to enhance the interactive capabilities of humanoid robots like Abel, aiming to develop solutions that seamlessly integrate robotic innovations with human needs.



Lorenzo Cominelli , holds a Master's Degree in Biomedical Engineering (2014) and a Ph.D. in Automation, Robotics, and Biomedical Engineering (2018) from the University of Pisa. Currently, he works as a Technologist at the Dipartimento di Ingegneria dell'Informazione (DII) in collaboration with the "E. Piaggio" Research Center. His research focuses on AI and Cognitive Systems for Social Robotics, with a particular interest in emotions and their influence on robots' decision-making. Since 2021, he has been developing Abel, a hyper-realistic humanoid robot with expressive emotional capabilities.

Enzo Pasquale Scilingo , PhD, is a Full Professor in Electronic and Information Bioengineering at the University of Pisa. He has several teaching activities and he is the supervisor of several PhD students. He coordinated the European projects "PSYCHE" and "NEVERMIND", and he is currently coordinating the H2020 European project "POTION". His main research interests are in wearable monitoring systems, human-computer interfaces, biomedical and biomechanical signal processing, modeling, control, and instrumentation. He is the author of more than 200 papers in peer-reviewed journals, contributions to international conferences, and chapters in international books.