An Incremental Learning Framework for Human-like Redundancy Optimization of Anthropomorphic Manipulators

Hang Su[1], *Member, IEEE*, Wen Qi[1], *Member, IEEE*, Yingbai Hu[2], Hamid Reza Karimi[3], *Senior Member, IEEE*, Giancarlo Ferrigno[1], *Senior Member, IEEE* and Elena De Momi[1], *Senior Member, IEEE*

***Abstract*— Recently, the human-like behavior on the anthropomorphic robot manipulator is increasingly accomplished by the kinematic model establishing the relationship of an anthropomorphic manipulator and human arm motions. Notably, the growth and broad availability of advanced data science techniques facilitate the imitation learning process in anthropomorphic robotics. However, the enormous data set causes the labeling and prediction burden. In this paper, the swivel motion reconstruction approach was applied to imitate human-like behavior using the kinematic mapping in robot redundancy. For the sake of efficient computing, a novel incremental learning framework that combines an incremental learning approach with a deep convolutional neural network (IN-DCNN) is proposed for fast and efficient learning. The algorithm exploits a novel approach to detect changes from human motion data streaming and then evolve its hierarchical representation of features. The incremental learning process can fine-tune the deep network only when model drifts detection mechanisms are triggered. Finally, we experimentally demonstrated this neural network's learning procedure and translated the trained human-like model to manage the redundancy optimization control of an anthropomorphic robot manipulator (LWR4+, KUKA, Germany). This approach can hold the anthropomorphic kinematic structure-based redundant robots. The experimental results showed that our architecture could not only enhance the regression accuracy but also significantly reduce the processing time of learning human motion data.**

I. INTRODUCTION

Human-like practice imitation and analysis have attracted increasing research attention in anthropomorphic robotics control over the past decades [1], [2]. It has been demonstrated that the human-like motion control of anthropomorphic manipulators is capable of enhancing the quality of Human-Robot Interaction (HRI) prominently in multiple areas, like industry and biomedical engineering purposes [3]–[5]. Especially for the anthropomorphic serial robot with human-like mechanical structures, for example, YuMi (ABB, Zurich, Switzerland), Justin robot (Institute of Robotics and Mechatronics, Wessling, Germany), and LWR4+ (KUKA, Augsburg, Germany) resembling human-like action in the kinematic level can be more social, cognitive and reasonable.

This work was supported by the European Commission Horizon 2020 research and innovation program, under the project SMARTsurg, grant agreement No. 732515.(Corresponding author: Wen Qi.)*

[1]Hang Su, Wen Qi, Giancarlo Ferrigno and Elena De Momi are with the Department of Electronics, Information and Bioengineering, Politecnico di Milano, 20133, Milan, Italy (e-mail: hang.su@polimi.it; wen.qi@polimi.it; giancarlo.ferrigno@polimi.it; elena.demomi@polimi.it).

[2]Yingbai Hu is with Department of Informatics, Technical University of Munich, Munich, 85748, Germany (e-mail: yingbai.hu@tum.de).

[3]Hamid Reza Karimi is with the Department of Mechanical Engineering, Politecnico di Milano, 20133 Milan, Italy (e-mail: hamidreza.karimi@polimi.it).

Many investigations had been delivered to implement human-like conduct presence on the robot's end-effector. An autonomous adaptation human-like control for reaching the target of hands-on surgical robots was completed by Beretta *et al.* in [6]. A human-like reaching motion had been proposed in [7] for robot-environment interactions. The hand pose's human-like path planning was achieved using a feed-forward Artificial Neural Network (ANN) in [8]. However, these contributions cannot control the whole-body of mimicking human-like behavior because they ignore the arm pose on manipulators.

For solving these limitations, the elbow swivel motion angle of the human arm is defined, shown in Fig. 1, to achieve whole-body human-like motion. The wrist-elbow-in-line approach is implemented to project human-like kinematics resolution for mapping the demonstrated human elbow angle based on the real robot [9]. The elbow swivel angle is redefined by mapping the modeled human-like swivel motion to the Yumi robot to imitate human behavior [10], [11]. The relationship of the hand pose and the elbow swivel angle is investigated for improving the human-like model. Moreover, the nonlinear regression relation between elbow swivel angles and hand poses achieves human-like deliveries [12]. A categorical mathematical model cannot analyze the relationship between end-effector configuration and elbow swivel angle. Recently, both regression errors and prediction time are tested based on different regression models, such as curve fitting (CF), artificial neural network (ANN) [13], and deep learning (DL) approach [14]). The comparison results show the CF model cannot map the non-stationary relationship of them. Although the feed-forward neural network (FFNN) algorithm with a single hidden layer was utilized for accuracy enhancement, it limits the real-time prediction. The development of the advanced data science techniques, such as machine learning (ML) and intelligence computation, facilitates imitation learning in anthropomorphic robotics [15]. However, the high computational burden is imposed by huge data streams.

Due to the overfitting of neural network (NN) models and low-speed computation, they cannot satisfy efficient learning requirements in a nonstationary environment, the big data processing techniques are widely used for making the robot imitating human-like behavior. The main application areas are the evolution of computing, the Internet of Things

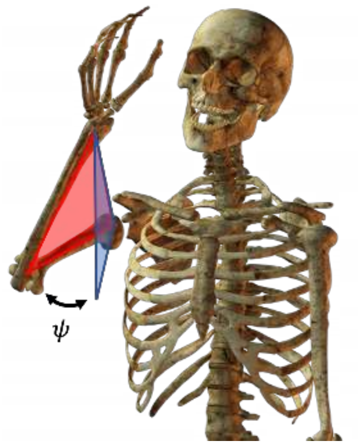


Fig. 1: Elbow swivel angle.

(IoT), humanoid robots, and HRI. In [16], a deep convolutional neural network (DCNN) is utilized based non-linear modeling approach for fast computation, noise robustness, and accuracy enhancement. However, most of the traditional methods cannot satisfy the fast-speed calculation with high accuracy in processing large-volume data.

Since the traditional continuous learning procedure is costly and time-consuming, developing a fast evolution method in the dynamic human-like learning procedure is necessary. The incremental learning approach [17] provides an efficient solution for model evolution which utilizes the notion of "concept drift" [18]. The model will be updated for adapting to the changes (only) when the change is detected.

This paper proposes an incremental learning framework based on the DCNN approach (IN-DCNN) for fast online human-like motion learning. This architecture aims to improve the existing DCNN model only when the drift detection mechanism is triggered. It incorporates the online learning regression model, along with an adaptive detection unit. The designed DCNN structure investigates the nonlinear relation between the human hand pose and the human swivel angle for improving the ability of regression analysis. Meanwhile, it is expected for accuracy enhancement and fast computation. In the experiments, the performance of the build IN-DCNN model is demonstrated by several human motion trajectory datasets, and results show that the proposed IN-DCNN method obtains the lowest Root Mean Square Error (RMSE) and computational time compared with other existing methods. Then, it is translated to manage whole-body human-like kinematic control of the 7 DoFs anthropomorphic robot arm (LWR4+, KUKA, Germany). The presented IN-DCNN framework includes the following contributions:

- A new DCNN architecture is designed to build the human motion model for achieving fast computation.
- An incremental learning framework is proposed for efficient human motion learning by utilizing "concept drift ". It adopts a non-parametric dynamic threshold method to detect drifts over streams data. When a change is detected in the data stream, the previous DCNN model will be updated.
- A decoupled control framework is utilized to achieve the whole-body human-like kinematic control of the robot manipulator during the tracking tasks.

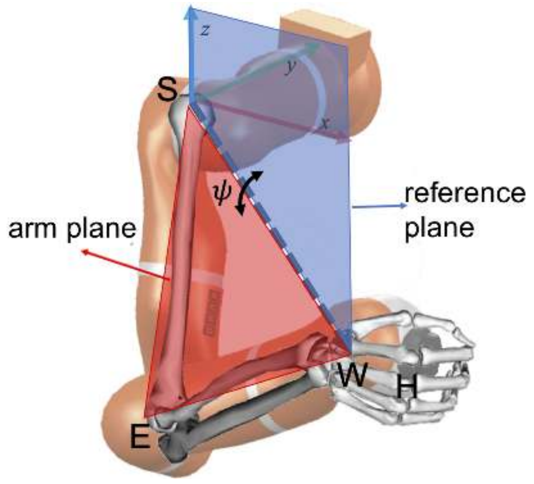


Fig. 2: Human-like kinematic mapping on the KUKA anthropomorphic manipulator. The coordination positions are labeled by elbow (E), hand (H), wrist (W), and shoulder (S), separately. The elbow swivel angle ψ is denoted by calulating the relationship of reference plane and arm plane.

The following sections organize the paper. Section II describes the details of the proposed IN-DCNN model and the designed hardware system. Section III presents the comparison experiment for proving the regression performance between IN-DCNN and other approaches. It also demonstrates the real-time prediction results by adopting the KUKA anthropomorphic manipulator. Finally, Section IV concludes and discusses further work.

II. METHODOLOGY

To analyze the swivel motion characteristics during manipulated tasks, it needs to build a human arm's kinematic model. Then, we present the online training framework based on the IN-DCNN learning method. Finally, the decoupled control approach is adopted to transfer the established human-like motion model.

A. *Human upper limb kinematic modeling*

Fig. 2 shows the human-like motion model building procedure for achieving a human-like kinematic mapping on the robot. Hence, the 7 DoFs rigid kinematic chain should be modeled based on the human arm (see Fig. 3). The expression of the elbow swivel angle ψ has been presented in [16]. The forward kinematics function provides hand poses by using Denavit-Hartenberg (D-H) parameters, and then the geometry relation helps to compute the joint angles $(q_i, i = 1, 2, \cdots, 7)$ [19]. Therefore, the swivel angle ψ can be denoted by the reference plane and the arm plane [13].

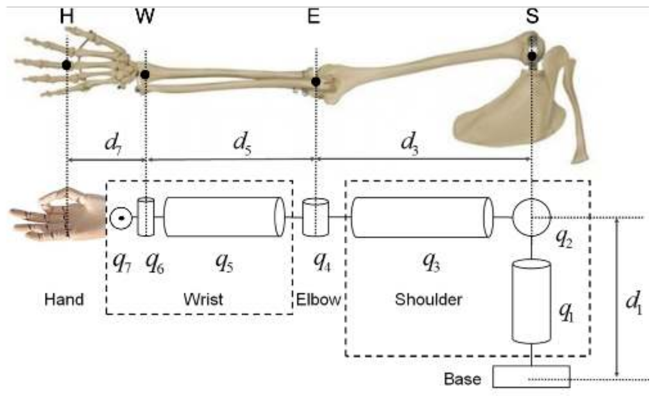


Fig. 3: The kinematic chain of a human upper limb, where the joint positions is $q_i, i = 1, \cdots, 7$ and the link lengths of each segment is $d_j, j = 1, 3, 5, 7$, separately.

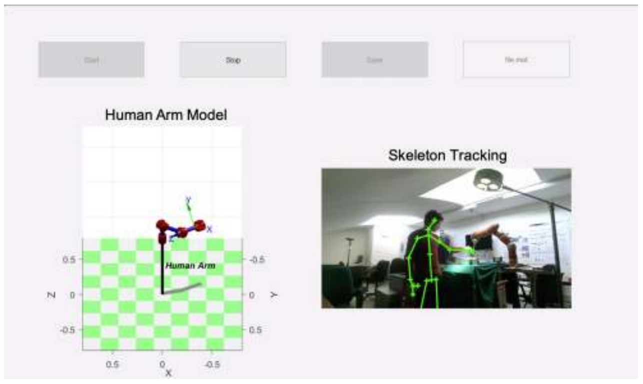


Fig. 4: Skeleton tracking and modeling interface.

B. Acquisition and preprocessing of human motion data

An acquisition software interface (MATLAB 2018b) is created using KINECT V2 (Microsoft, USA) device for collecting the human motion data, namely, swivel angles and skeleton (see Fig. 4) [20]. The "Start" button can activate the visual system, which is at the skeleton viewer's upward side. The description of the data acquisition has been discussed in [13].

The work aims to establish a DCNN-based regression model to map the 6-D hand pose $\mathbf{x} = [x, y, z, \theta_x, \theta_y, \theta_z]$ to the 1-D elbow swivel angle ψ, where x,y, and z are positions in the Cartesian coordinate system and θ_x, θ_y, and θ_z are Euler angles.

Since the DCNN model is the basic structure in the proposed incremental DCNN framework and convolutive layers should be applied to filter the input data, they are good for pixels or sequences of the homogenous dataset [21]. Therefore, the raw input $\mathbf{x} \in \mathbb{R}^{6\times1}$ should be extended into a homogenous matrix $\mathbf{x}^* \in \mathbb{R}^{6\times3}$ as follows:

$$\mathbf{x}^* = [\mathbf{x}; \mathbf{x} - \bar{\mathbf{x}}; \frac{\mathbf{x} - \bar{\mathbf{x}}}{\sigma_{\mathbf{x}}}] \tag{1}$$

where $\sigma_{\mathbf{x}}$ and $\bar{\mathbf{x}}$ are the standard deviation and average of $\mathbf{x}$, respectively.

C. The designed DCNN Structure

Although the ANN-based approaches demonstrated a promising performance, their complicated function structure will often cause them to be overfitting, underfitting, and time-consuming [13]. The DCNN-based system can solve these problems using batch normalization, ReLU activation function, and dropout layer because they are generally known for fast computation and resolving the overfitting problem [16].

The DCNN model consists of two convolutional modules, a dropout layer, and a full connection layer. Each convolutional module has two dimensions (2-D) convolutive map, Rectified Linear Unit (ReLU) activation function, batch normalization (BN) (see Fig. 5). Since the tensor representation should use each heterogeneous object for capturing the complex relations, the DCNN model is devised to learn features. The kernel tensor maps the six-order input ($\mathbf{x}^*$) K_m and bias b_m as follows:

$$F^o = f_c(\mathbf{x}^* \otimes K_m + b_m), m = 1, 2 \tag{2}$$

where f_c stands for the non-linear function and $\otimes$ represents the convolution operation. Then, the obtained features vector F^o will be normalized by

$$\begin{aligned} \mu_B &= \frac{1}{n}\sum_{1}^{n} F_i^o \\ \sigma_B^2 &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(F_i^o - \mu_B)^2} \\ \hat{F_i^o} &= \frac{F_i^o - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \\ y_i^B &\leftarrow \gamma \hat{F_i^o} + \beta \equiv BN_{\gamma,\beta}(F_i^o) \end{aligned} \tag{3}$$

The obtained output y_i^B is filtered by the ReLU function as:

$$y_i^R = f_r(y_i^B) = \max(0, w^\top y_i^B + b) \tag{4}$$

where w means weights and b is the bias. In this paper, the 2-D convolutive layer takes the output by filtering the input matrix with eight kernels of 2×2 dissensions.

The dropout layer aims to avoid the overfitting drawback by randomly dropping each neuron and sampling an ensemble of thinned deep architectures. The following equations describe the feed-forward procedure:

$$\begin{aligned} & s_i^{(\mathrm{h})} \sim \text{Bernoulli}(p) \\ & \widetilde{\mathbf{a}}^{(t)} = \mathbf{s}^{(\mathrm{h})} * \mathbf{a}(t) \\ & y_i^R(t+1) = \mathbf{w}_i(t+1)\widetilde{\mathbf{a}}(t) + b_i(t+1) \\ & a_i^{(t+1)} = f_d(y_i^R(t+1)) \end{aligned} \tag{5}$$

The input y_i^R are processed by the dropout network, where $\mathbf{w}$ and b denote the weights and bias in the hidden layer, respectively. The Bernoulli distribution $s^{(\mathrm{h})}$ is the mask matrix of each element, while $\widetilde{\mathbf{a}}^{(t)}$ is the masked output vector. The loss only passes based on the selected structure in the backpropagation [22]. Finally, the DCNN structure

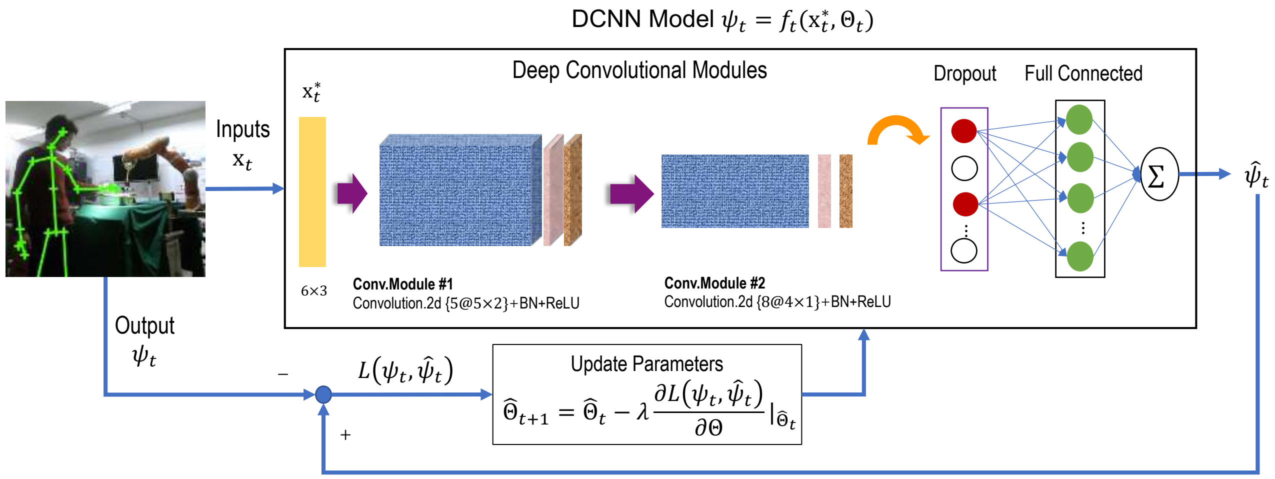

DCNN Model $\psi_t = f_t(x_t^*, \Theta_t)$
Deep Convolutional Modules
Inputs x_t
x_t^*
6×3
Conv.Module #1
Convolution.2d {5@5×2}+BN+ReLU
Conv.Module #2
Convolution.2d {8@4×1}+BN+ReLU
Dropout
Full Connected
Σ
$\hat{\psi}_t$
Output ψ_t
−
+
$L(\psi_t, \hat{\psi}_t)$
Update Parameters
$\hat{\Theta}_{t+1} = \hat{\Theta}_t - \lambda \frac{\partial L(\psi_t, \hat{\psi}_t)}{\partial \Theta}|_{\hat{\Theta}_t}$

Fig. 5: The schematic diagram of DCNN-based online learning framework. The parameters of the DCNN model $\psi = f(\mathbf{x}, \Theta)$ are modified in gradient descent learning way when the loss function L measures the discrepancy between the observed ψ_t and predicted $\hat{\psi}_t$.

captures new knowledge by reusing the idle subnetworks. The applied dropout layer (with 0.3 percentage) aims to avoid the overfitting problem and time-consuming tasks. The adaptive moment estimation optimizer (Adam) function is adopted with a 0.001 learning rate. We also add the dropout layer for avoiding overfitting. The drop factor and drop period are 0.001 and 500, respectively. Although the built DCNN regression model predicts the hand poses with high accuracy, it cannot learn new information in a dynamic environment.

D. The proposed IN-DCNN framework

Although the three components of the DCNN model are architecturally elaborated enough to obtain heterogeneous characteristics, the fast-growing data stream's dynamic characteristic is still a challenge for HAR [23]. Meanwhile, the DCNN model uses multiple layers because it is safe from the gradient vanishing problem. However, the DCNN regression model is demanded to deal with concept drifts in the real-world scenarios where a time-varying behavior characterizes the data streaming [24]. Hence, we propose the incremental DCNN (IN-DCNN) framework to solve this problem, which only updates the parameters in a fully connected layer for fast computation. The IN-DCNN model consists of two stages, namely the initial increment computation and the parameters re-training [25]. Each of them should include the convergence analysis procedure. In order to avoid losing the current information in the tensor space, the full connection network's parameters adopt a new loss function while the parameters were re-training step capture new information to train the previous knowledge by fine-tuning method [26].

1) Convergence Analysis: We adopt the supervised machine learning mechanism [1] to evaluate the function of the IN-DCNN framework. In Fig. 5, the time-varying reconstructed inputs $\mathbf{x}_t^*$ provide new information of the built classifier $\hat{\psi}_t = f_t(\mathbf{x}_t^*, \Theta_t)$. The online learning IN-DCNN model is designed to modify the overall parameters continuously set $\Theta_t = \{\mathbf{W}_t, \mathbf{b}_t\}$, where $\mathbf{W}$ and $\mathbf{b}$ denote all of the weights and bias of DCNN classifier, respectively.

The purpose of nonlinear modeling is to search the optimal features by calculating the minimum least squares.

$$\begin{aligned}\Theta_t &= \underset{\Theta}{\text{argmin}} \sum_{t=1}^{N} \left(\hat{\psi}_t - \psi_t\right)^2 \\ &= \underset{\Theta}{\text{argmin}} \|\hat{\psi}_t - \psi_t\|_2^2\end{aligned} \tag{6}$$

The performance of DCNN model is evaluated by Root Mean Square Error (RMSE) ε_t at time t as follows:

$$\varepsilon_t = \sqrt{\sum_{t=1}^{N} (\frac{\hat{\psi} - \psi}{t})^2} \tag{7}$$

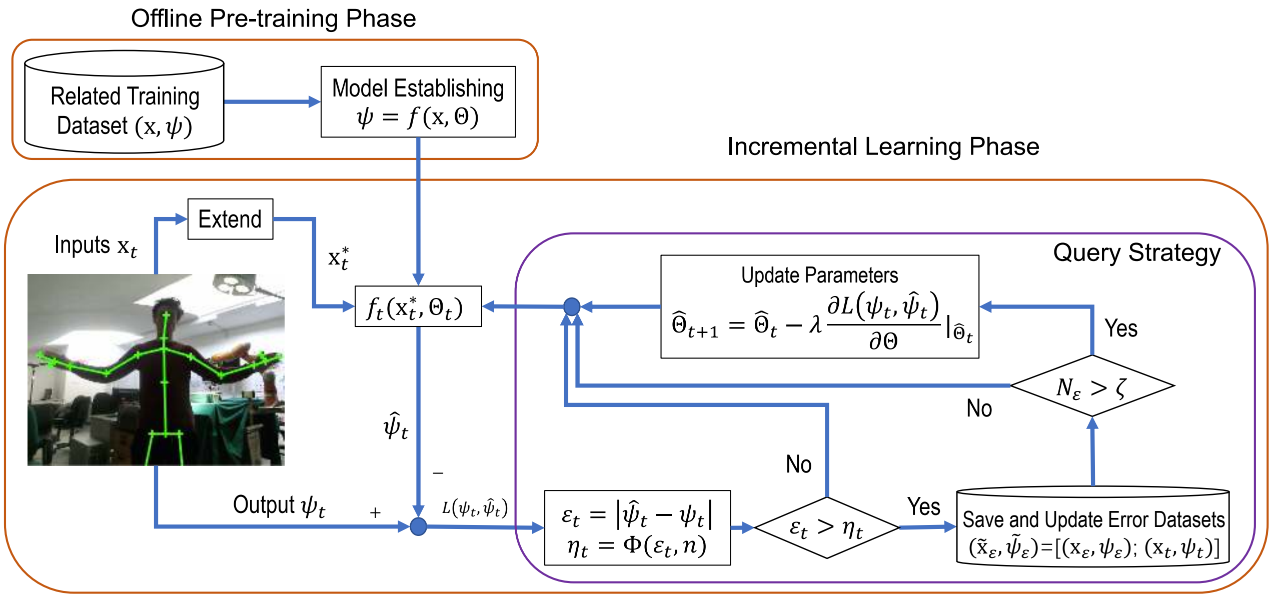
where N is the length of the whole dataset, a lower RMSE value is expected. IN-DCNN algorithm aims to update the parameters set Θ_t when new couple $(\mathbf{x}_t, \psi_t)$ is available at time t continuously as follows:

$$\hat{\Theta}_{t+1} = \hat{\Theta}_t - \lambda \frac{\partial L\left(\psi_t, \hat{\psi}_t\right)}{\partial \Theta}\Bigg|_{\hat{\Theta}_t} \tag{8}$$

Where λ denotes the learning rate of this deep neural network, and L is the loss function, which measures the squared error between the identified swivel angle $\hat{\psi}$ and the observed swivel angle ψ (see Eq. 9). It is a typical gradient descent learning mechanism for passive solutions [27].

$$\Delta L = \frac{1}{2} \psi_t^\top \Theta \hat{\psi}_t = f_t(\mathbf{x}_t^*, \Theta_t) - \psi_t \tag{9}$$

[1]. In a supervised learning model, the algorithm learns on a labeled dataset, providing an answer key that the algorithm can use to evaluate its accuracy on training data. In contrast, an unsupervised model provides unlabeled data that the algorithm tries to make sense of by extracting features and patterns on its own.



Offline Pre-training Phase
Related Training Dataset (x, ψ)
Model Establishing $\psi = f(x, \Theta)$
Incremental Learning Phase
Inputs x_t
Extend
x_t^*
$f_t(x_t^*, \Theta_t)$
Query Strategy
Update Parameters
$\hat{\Theta}_{t+1} = \hat{\Theta}_t - \lambda \frac{\partial L(\psi_t, \hat{\psi}_t)}{\partial \Theta}|_{\hat{\Theta}_t}$
Yes
$N_\varepsilon > \zeta$
No
$\hat{\psi}_t$
No
−
Output ψ_t
+
$L(\psi_t, \hat{\psi}_t)$
$\varepsilon_t = |\hat{\psi}_t - \psi_t|$
$\eta_t = \Phi(\varepsilon_t, n)$
$\varepsilon_t > \eta_t$
Yes
Save and Update Error Datasets
$(\tilde{x}_\varepsilon, \tilde{\psi}_\varepsilon)$=[$(x_\varepsilon, \psi_\varepsilon)$; (x_t, ψ_t)]

Fig. 6: The schematic diagram of the DCNN-based incremental learning (IN-DCNN) framework. The whole parameters of DCNN model (structure described in fig. 5) will be updated when the loss function $L(\psi_t, \hat{\psi}_t)$ measuring the absolute error $|\psi_t - \hat{\psi}_t|$ is inspected by the query strategy. When $\varepsilon_t > \eta_t$, the new couple $(\mathbf{x}_t, \psi_t)$ will be saved into the storage space of error dataset $(\tilde{x}_\varepsilon, \tilde{\psi}_\varepsilon)$. The previous DCNN model evolution: The following condition is to judge whether the length of the error dataset N_e is larger than the threshold ζ. If both of the conditions are satisfied, it will start to modify the DCNN model's parameters.

The difference $\hat{\Theta}_{t+1} - \hat{\Theta}_t$ should be close to the final value to improve efficiency by reducing the re-training steps. The L2-norm between $\hat{\psi}$ and ψ offers the adaption information to evaluate the performance of preservation [28]. The deformation loss function L_o is

$$L_o = \frac{1}{2} \left\| f_t(\mathbf{x}_t^*, \Theta_t) - -\psi_t \right\|_2^2 \tag{10}$$

Hence, the increment loss component L is

$$\begin{aligned} L &= \Delta L + L_o \\ &= \frac{1}{2} \Delta y^T \Theta \Delta y + \frac{1}{2} \left\| f_t(\mathbf{x}_t^*, \Theta_t) - -\psi_t \right\|_2^2 \end{aligned} \tag{11}$$

When a new couple $(\mathbf{x}_t^*, \psi_t)$ is detected, the query strategy is triggered to evaluate whether it needs to update the parameters set Θ by comparing the difference $\varepsilon_t - \eta_t$. Where η_t is the time-varying error threshold. The increment computation problem is transformed to minimize the increment loss component L as:

$$\begin{aligned} \frac{\partial L}{\partial \Theta} &= \frac{\partial}{\partial \Theta} \left(\frac{1}{2} \Delta y^T \Theta \Delta y + \frac{1}{2} \left\| f_t(\mathbf{x}_t^*, \Theta_t) - -\psi_t \right\|_2^2 \right) \\ &= 0 \end{aligned} \tag{12}$$

To avoid computational redundancy and enhance accuracy, a query strategy is proposed in IN-DCNN architecture (see Fig. 6). It includes two judgment mechanisms. The absolute error $\varepsilon_t = |\hat{\psi}_t - \psi_t|$ is measured every time by comparing with the adaptive threshold η_t, which is designed for drifts detection. η_t is the ξ times Quantile of the continuous updated absolute errors sequence ε_t as:

$$\eta_t = \Phi(\varepsilon_t, n) = \xi \times (n-1)^{th} [\varepsilon_o; \varepsilon_t] \tag{13}$$

where ε_o is the initial error and ε_t is the current value, respectively. In this paper, $\xi = 0.4$. The current couple $(\mathbf{x}_t, \psi_t)$ will be saved into the training dataset if $\varepsilon_t > \eta_t$. Meanwhile, the query mechanism continuously counts the number N_e of the increased error sequence $(\tilde{x}_\varepsilon, \tilde{\psi}_\varepsilon) = [(x_\varepsilon, \psi_\varepsilon); (x_t, \psi_t)]$. When $N_\varepsilon > \zeta$, the parameters of the previous DCNN model will be updated. ζ is the threshold of the length of error dataset. In this paper, we set $\zeta = 500$ due to the four testing datasets have 1500 samples. The query unit will be reset after the gradient descent is activated.

2) *Parameters Re-training:* Although the proposed IN-DCNN algorithm only updates the full connection layer parameters, it needs to make the IN-DCNN model capturing the aggregate data's intrinsic features by the incremental dropout re-training algorithm. The weights can be classified into three subsets with two thresholds (i.e., η_1 and η_2). The assigned probability $\mathcal{P}_j, j = 1, 2, 3$ of each subset subjects to the Bernoulli distribution. The incremental knowledge can be saved into the idle neurons with a large probability [28]. Thus, they are easy to be re-trained in the dropout updating phase.

The norm of N-order weights could be calculated by:

$$\|\mathrm{w}\| = w \otimes w = \sum_{j_1=1}^{t_1} \cdots \sum_{j_n-1}^{t_{N-1}} w_{j_1 j_2 \ldots j_{N-1}}^2 \tag{14}$$

And the subspaces' weight are divided into the following three sections:

$$\begin{aligned} \mathbf{W}_1 &: 0 < \|\mathrm{w}\| < \eta_1 \\ \mathbf{W}_2 &: \eta_1 \leq \|\mathrm{w}\| < \eta_2 \\ \mathbf{W}_3 &: \|\mathrm{w}\| \geq \eta_2 \end{aligned} \tag{15}$$

In the incremental learning procedure, each subspace plays a different role. Eq. 15 compute the probability of each subset. Most of the old information is preserved in the neurons of subspace $\mathbf{W}_2$, while $\mathbf{W}_1$ is effected on the features learning. Hence, we update the whole parameters by the initial parameter, $\Delta\mathbf{W}$ increment as follows:

$$\mathbf{W}_{all} = \mathbf{W} + \Delta\mathbf{W} * M \tag{16}$$

Then, $Y_o = f(\mathbf{W}_{all} \odot \mathbf{x} \otimes M + b)$ is the tensor full connected layer's outputs. Hence, the backpropagation of incremental training algorithm consists of three steps as follows:

1) compute the incremental changes of loss function for each neuron k, namely

$$\delta L^k = \frac{\partial L}{\partial \Theta^k} = \frac{\partial}{\partial \Theta}\frac{1}{2}(f_t^k(\mathbf{x}_t^*, \Theta_t) - \psi_t^k)^2$$

2) the incremented changes of full connection layer can be calculated by

$$\delta L^k = (\mathbf{W}^k)^\top \odot L^{k+1} \otimes Y_o' \otimes M$$

3) obtain the incremental weights and bias by

$$\Delta\mathbf{W}^k = M^k \odot L^{k+1}, \Delta b^k = L^{k+1}$$

E. Robotic human-like kinematic control

For the robot manipulator control, a velocity-based controller is investigated in this paper to track the desired target pose $\boldsymbol{X_d}$ using the human-like swivel angle ψ_d. The desired control input and the joint velocity $\dot{\boldsymbol{q}}$, can be obtained using the mapping from the end-effector's velocity $\dot{\boldsymbol{X}}$, which is represented as:

$$\dot{\boldsymbol{X}} = \boldsymbol{J}\dot{\boldsymbol{q}}_T \tag{17}$$

where $\dot{\boldsymbol{q}}_T$ determines the required joint velocities to achieve the performing task. $\dot{\boldsymbol{q}}_N$ is defined as the joint motion generated in the Null-space of the corresponding performing task. The Jacobian matrix $\boldsymbol{J}(\boldsymbol{q}) \in \mathbb{R}^{6\times 7}$ is acquired from the manipulator tool pose to the base of the manipulator. This formulation demonstrates the projection relation between joint-space and task-space. The given pose of the joint-space configuration solution is infinite due to the redundant structure of the manipulator. Null-space projection is an efficient way to determine redundancy resolution, which can be expressed as:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^+\dot{\boldsymbol{X}} + \left(I - \boldsymbol{J}^+\boldsymbol{J}\right)\boldsymbol{J_E}^+\dot{\psi}\boldsymbol{u}_\psi \tag{18}$$

where, the Jacobian matrix $\boldsymbol{J_E} \in \mathbb{R}^{3\times 4}$ is camputed from the manipulator's elbow to the manipulator base. $\dot{\psi}$ is the velocity of the swivel angle ψ [13]. This presents the relationship of the joint-space, task space and the elbow position. The velocity direction vector of the elbow joint, $\boldsymbol{u}_\psi \in \mathbb{R}^{3\times 1}$, can also be viewed as the direction vector of the elbow swivel motion, represented as follows:

$$\boldsymbol{u}_\psi = \frac{\overrightarrow{\boldsymbol{SE}} \times \overrightarrow{\boldsymbol{EW}}}{\|\overrightarrow{\boldsymbol{SE}} \times \overrightarrow{\boldsymbol{EW}}\|} \tag{19}$$

$\overrightarrow{\boldsymbol{SE}}$ represents the vector from the shoulder of the manipulator to the elbow of the manipulator. $\overrightarrow{\boldsymbol{EW}}$ is the vector from the manipulator's to the wrist of the manipulator (see Fig. 2). This also works for the desired variables, as follows:

$$\dot{\boldsymbol{q}}_d = \boldsymbol{J}^+\dot{\boldsymbol{X}}_d + \left(I - \boldsymbol{J}^+\boldsymbol{J}\right)\boldsymbol{J_E}^+\dot{\psi}_d\boldsymbol{u}_\psi \tag{20}$$

In this work, the manipulator is assumed far away from its singularity and the pseudo-inverse of the Jacobian matrix [29], $\boldsymbol{J}^+$, exists. The sketch of the corresponding decoupled kinematic controller is demonstrated in Fig. 7.

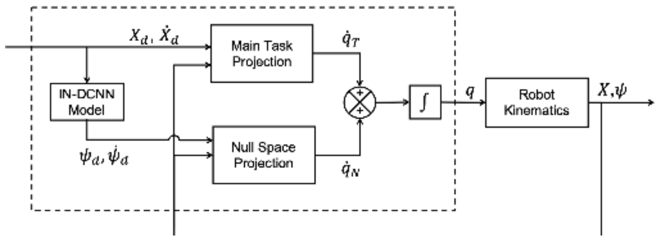


Fig. 7: A sketch of the human-like kinematic controller.

III. Experiment and Demonstration

The performance of the proposed IN-DCNN structure is verified by conducting the following two experiments. Five trajectory datasets (one for training and four for testing) are collected. The detailed definition of the data collection procedure and task performance have been depicted in [13]. Both DL and ML approaches are adopted, such as long-short-terms memory (LSTM), DCNN, and ANN-based algorithms. Especially, cascade-forward neural networks (CFNN) and FFNN (with 20 neurons in the hidden layer) are used to build the ANN-based models [30]. Quantitative evaluations are used to test the ability of the IN-DCNN framework for online prediction. All of the methods are implemented and compared in the same software (MATLAB 2018b) and hardware platform based on Intel(R) i7 Core 2.80 GHz CPU and 16.0 GB RAM.

A. Performance comparisons of IN-DCNN modeling

After building the regression models on the training dataset (2000 samples), their performance will be evaluated on the four trajectory groups (each has 1500 samples). Table I shows the comparison results of RMSE and cumulative predictive time *ct* among the built models. To avoid overfitting or underfitting, we run the results over ten times. The LSTM architecture is designed with an LSTM layer (150 neurons) and a dropout layer (0.3 percentage). The Adam estimation optimizer is used. The parameters are set as follows: the learning rate is 0.01, the minimum batch size is 50, the drop factor is 0.02, and the drop period is 5.

TABLE I: The comparative results of IN-DCNN, DCNN, LSTM, FFNN, and CFNN models on the four trajectories.

Model	Parameters	Datasets Number			
		Task 1	Task 2	Task 3	Task 4
IN-DCNN	RMSE (rad)	**0.1820** $\pm$ **0.0278**	**0.1540** $\pm$ **0.0223**	**0.2187** $\pm$ **0.0249**	**0.2017** $\pm$ **0.0175**
	$\sum ct$ (s)	**23.79** $\pm$ 0.15	**22.73** $\pm$ 0.19	**27.91** $\pm$ 0.14	**26.64** $\pm$ 0.13
DCNN	RMSE (rad)	0.1866 $\pm$ 0.0562	0.1761 $\pm$ 0.0623	0.2292 $\pm$ 0.0422	0.2026 $\pm$ 0.0485
	$\sum ct$ (s)	2426.2 $\pm$ 9.04	2440.0 $\pm$ 9.23	2509.4 $\pm$ 9.77	2450.6 $\pm$ 9.10
LSTM	RMSE (rad)	1.2714 $\pm$ 0.0562	1.3452 $\pm$ 0.0623	1.2980 $\pm$ 0.0607	1.3575 $\pm$ 0.0579
	$\sum ct$ (s)	1571.0 $\pm$ 5.14	1576.0 $\pm$ 5.11	1882.3 $\pm$ 5.13	1594.9 $\pm$ 5.10
FFNN [31] [13]	RMSE (rad)	0.1848 $\pm$ 0.1677	0.1631 $\pm$ 0.1331	0.2172 $\pm$ 0.1468	0.2019 $\pm$ 0.1655
	$\sum ct$ (s)	195.70 $\pm$ 2.21	191.11 $\pm$ 2.34	201.44 $\pm$ 3.09	190.34 $\pm$ 2.74
CFNN [32]	RMSE (rad)	0.1948 $\pm$ 0.1319	0.1762 $\pm$ 0.1205	0.2271 $\pm$ 0.1327	0.1994 $\pm$ 0.1244
	$\sum ct$ (s)	204.81 $\pm$ 3.22	198.70 $\pm$ 3.40	198.94 $\pm$ 3.11	196.21 $\pm$ 3.48

As expected, the proposed IN-DCNN framework is the fastest method in the online regression scheme. It only needs around 25 seconds to predict all of the results in each trajectory, while the other four models spend a colossal time. Meanwhile, the IN-DCNN model obtains the lowest RMSE even if they are close to that of the DCNN model. Furthermore, the obtained lowest standard deviation of RMSE proves that the IN-DCNN algorithm is a robust method.

Fig. 8 displays four examples of online RMSE values comparing among the IN-DCNN, DCNN, LSTM, FFNN, and CFNN methods on the four trajectories. The red curves are the results obtained by the IN-DCNN method. The above results prove that the IN-DCNN method is capable of achieving high accuracy and fast computation.

B. Demonstration

After validating the performance in the IN-DCNN based human-like model training process, the human-like redundancy optimization using the trained model is demonstrated on an anthropomorphic KUKA robot. This requires the robot to perform the collected human motion in an autonomous way [13]. With the same end-effector task, the swivel motion of the elbow should be in a similar way. The anthropomorphic robot manipulator pose is computed using an interpolation algorithm and used as the DNN model's input. The DCNN regression model can predict the human-like swivel motion angle. Meanwhile, the results are updated based on the incremental learning procedure. The joints configuration is obtained by utilizing the decoupled control strategy. Fig. 9 illustrates the human kinematics strategies when executing tracking tasks, which achieve human-like arm posture prediction.

IV. Conclusion and Future Work

In this paper, an incremental learning framework based on the DCNN method (IN-DCNN) with a query strategy is proposed for the human-like system. This scheme can be implemented on time-varying big data streams. The experimental results show the IN-DCNN algorithm achieving online regression prediction for accuracy enhancement and fast computation. Meanwhile, the proposed IN-DCNN method obtains the lowest RMSE and computational time comparing with the other models (i.e., IN-DCNN, DCNN, LSTM, FFNN, and CFNN). Finally, the trained human-like kinematic model is utilized to manage the redundancy control of a 7 DoFs anthropomorphic robot manipulator for validation. In the future, remote minimally invasive surgery (MIS) could be further developed through the advance of communication technology and could computing. To improve the realtime performance of remote MIS, computational cost should be as low as possible. Therefore, the proposed IN-DCNN method can also save costs in the area of cloud computing due to the high computational ability of IN-DCNN. Although human-like reaching motion could enhance the equality of human-robot interaction; it is still challenging to deal with the mapping between the human upper limb and the manipulator. Therefore, it is necessary to investigate more efficient human upper limb kinematic models for achieving a human-like kinematic mapping on the robot, considering joint limits and other various performance indices, such as manipulability, repetitive motion performance index, etc.. Besides, human operators should also perform a non-singular trajectory in order to avoid the singularity case. In future work, we will further investigate the performance of the proposed method in the presence of noisy trajectories.

References

[1] S. Coradeschi, H. Ishiguro, M. Asada, S. C. Shapiro, M. Thielscher, C. Breazeal, M. J. Mataric, and H. Ishida, "Human-inspired robots," *IEEE Intelligent Systems*, vol. 21, no. 4, pp. 74–85, 2006.

[2] H. Huang, C. Yang, and C. L. P. Chen, "Optimal robot-environment interaction under broad fuzzy neural adaptive control," *IEEE Transactions on Cybernetics*, vol. PP, no. 99, pp. 1–12, 2020.

[3] F. Ficuciello, L. Villani, and B. Siciliano, "Variable impedance control of redundant manipulators for intuitive human–robot physical interac-

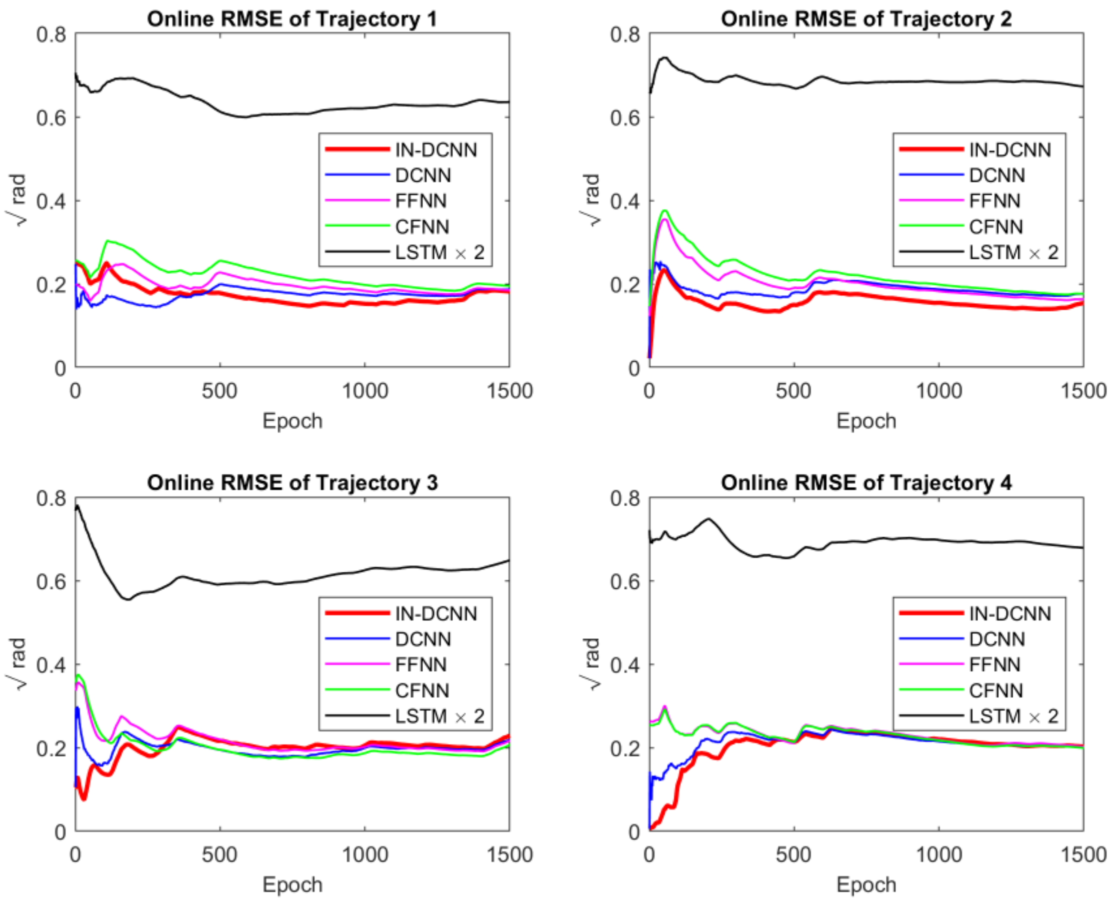


Fig. 8: The online RMSE values of IN-DCNN, DCNN, LSTM, FFNN, and CFNN methods on the four trajectories. Where the RMSE values computed by LSTM model are divided by two.

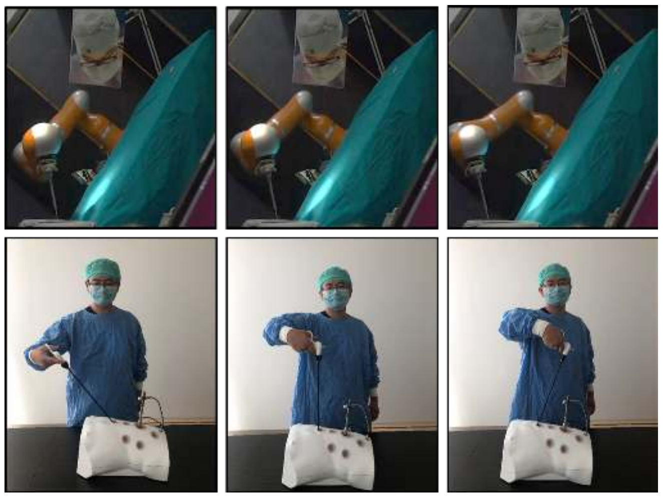


Fig. 9: The demonstration of human-like motion prediction.

tion," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 850–863, 2015.

[4] Y. Tang, X. Xing, H. R. Karimi, L. Kocarev, and J. Kurths, "Tracking control of networked multi-agent systems under new characterizations of impulses and its applications in robotic systems," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 2, pp. 1299–1307, 2016.

[5] H. Huang, T. Zhang, C. Yang, and C. L. P. Chen, "Motor learning and generalization using broad learning adaptive neural control," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 10, pp. 8608–8617, 2020.

[6] E. Beretta, E. De Momi, F. Rodriguez y Baena, and G. Ferrigno, "Adaptive hands-on control for reaching and targeting tasks in surgery," *International Journal of Advanced Robotic Systems*, vol. 12, no. 5, p. 50, 2015.

[7] A. Atawnih, D. Papageorgiou, and Z. Doulgeri, "Reaching for redundant arms with human-like motion and compliance properties," *Robotics and Autonomous Systems*, vol. 62, no. 12, pp. 1731–1741, 2014.

[8] E. De Momi, L. Kranendonk, M. Valenti, N. Enayati, and G. Ferrigno, "A neural network-based approach for trajectory planning in robot–human handover tasks," *Frontiers in Robotics and AI*, vol. 3, p. 34, 2016.

[9] W. Liu, D. Chen, and J. Steil, "Analytical inverse kinematics solver for anthropomorphic 7-dof redundant manipulators with human-like configuration constraints," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 1, pp. 63–79, 2017.

[10] A. M. Zanchettin, P. Rocco, L. Bascetta, I. Symeonidis, and S. Peldschus, "Kinematic analysis and synthesis of the human arm motion during a manipulation task," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2692–2697.

[11] H. Su, Y. Hu, H. R. Karimi, A. Knoll, G. Ferrigno, and E. De Momi, "Improved recurrent neural network-based manipulator control with remote center of motion constraints: Experimental results," *Neural Networks*, vol. 131, pp. 291–299, 2020.

[12] A. M. Zanchettin, L. Bascetta, and P. Rocco, "Acceptability of robotic manipulators in shared working environments through human-like redundancy resolution," *Applied ergonomics*, vol. 44, no. 6, pp. 982–989, 2013.

[13] H. Su, N. Enayati, L. Vantadori, A. Spinoglio, G. Ferrigno, and E. De Momi, "Online human-like redundancy optimization for tele-operated anthropomorphic manipulators," *International Journal of Advanced Robotic Systems*, vol. 15, no. 6, p. 1729881418814695, 2018.

[14] H. Su, W. Qi, C. Yang, A. Aliverti, G. Ferrigno, and E. De Momi, "Deep neural network approach in human-like redundancy optimization for anthropomorphic manipulators," *IEEE Access*, vol. 7, pp. 124 207–124 216, 2019.

[15] Z. Cao, C.-T. Lin, W. Ding, M.-H. Chen, C.-T. Li, and T.-P. Su, "Identifying ketamine responses in treatment-resistant depression using a wearable forehead eeg," *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 6, pp. 1668–1679, 2018.

[16] H. Su, W. Qi, C. Yang, A. Andrea, G. Ferrigno, and E. De Momi, "Deep neural network approach in human-like redundancy optimization for anthropomorphic manipulators," *IEEE ACCESS*, 2019.

[17] K. Shmelkov, C. Schmid, and K. Alahari, "Incremental learning of object detectors without catastrophic forgetting," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3400–3409.

[18] A. Haque, L. Khan, M. Baron, B. Thuraisingham, and C. Aggarwal, "Efficient handling of concept drift and concept evolution over stream data," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 2016, pp. 481–492.

[19] C. Gaz, F. Flacco, and A. De Luca, "Identifying the dynamic model used by the kuka lwr: A reverse engineering approach," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1386–1392.

[20] O. Patsadu, C. Nukoolkit, and B. Watanapa, "Human gesture recognition using kinect camera," in *2012 Ninth International Conference on Computer Science and Software Engineering (JCSSE)*. IEEE, 2012, pp. 28–32.

[21] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Information Fusion*, vol. 42, pp. 146–157, 2018.

[22] A. E. Essien and C. Giannetti, "A deep learning model for smart manufacturing using convolutional lstm neural network autoencoders," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2020.

[23] S. K. Kommuri, M. Defoort, H. R. Karimi, and K. C. Veluvolu, "A robust observer-based sensor fault-tolerant control for pmsm in electric vehicles," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 12, pp. 7671–7681, 2016.

[24] Z. Zheng, Y. Yang, X. Niu, H.-N. Dai, and Y. Zhou, "Wide and deep convolutional neural networks for electricity-theft detection to secure smart grids," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1606–1615, 2018.

[25] S. Wan and L. E. Banta, "Parameter incremental learning algorithm for neural networks," *IEEE Transactions on Neural Networks*, vol. 17, pp. p.1424–1438, 2006.

[26] F. Estrada-Solano, O. M. Caicedo, and N. L. S. D. Fonseca, "Nelly: Flow detection using incremental learning at the server-side of sdn-based data centers," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2019.

[27] T. Le, T. Nguyen, V. Nguyen, and D. Phung, "Dual space gradient descent for online learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 4583–4591.

[28] P. Li, Z. Chen, L. T. Yang, J. Gao, Q. Zhang, and J. Deen, "An incremental deep convolutional computation model for feature learning on industrial big data," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2018.

[29] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[30] S. Goyal and G. K. Goyal, "Cascade and feedforward backpropagation artificial neural networks models for prediction of sensory quality of instant coffee flavoured sterilized drink," *Canadian Journal on Artificial Intelligence, Machine Learning and Pattern Recognition*, vol. 2, no. 6, pp. 78–82, 2011.

[31] T.-Y. Kwok and D.-Y. Yeung, "Constructive algorithms for structure learning in feedforward neural networks for regression problems," *IEEE transactions on neural networks*, vol. 8, no. 3, pp. 630–645, 1997.

[32] A. Pwasong and S. Sathasivam, "A new hybrid quadratic regression and cascade forward backpropagation neural network," *Neurocomputing*, vol. 182, pp. 197–209, 2016.