IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Data Science & Statistical Learning | II Level Master

Advanced Machine Learning

UNIVERSITÀ DEGLI STUDI FIRENZE

Pose Analysis of Humanoid Robot Imitation Process Based on Improved MLP Network

Prof. *Giorgio Gnecco*

Gianmarco Santoro

27/12/2023

- **Aim** – make **NAO humanoid robot** imitate and **learn human behaviours**
- **How** – stability identification method based on improved Multi-Layer Perception (**MLP**):
  1. Human **joints positions** are collected by a Kinect
  2. Info is transformed into robot angles by **inverse kinematics** equations
  3. Angles are used in a deep neural network to **identify robot stability**
- **Test** – to identify **accuracy** on imitation result

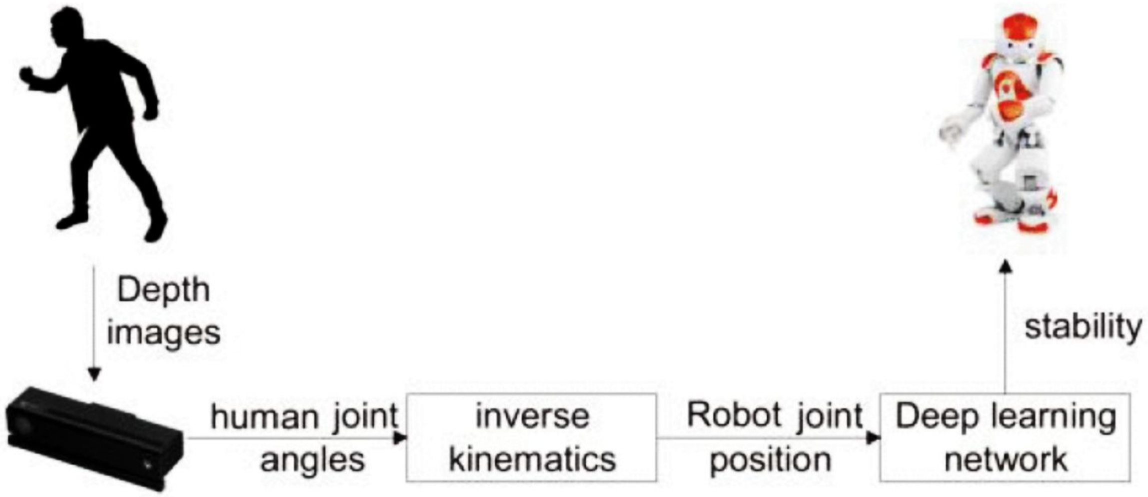


Image source: Paper*

- The **fall** of the robot directly leads to the **failure** of the imitation task

- To **ensure the stability** during the imitation, a deep neural network which **consider both arms and legs** of robot is used

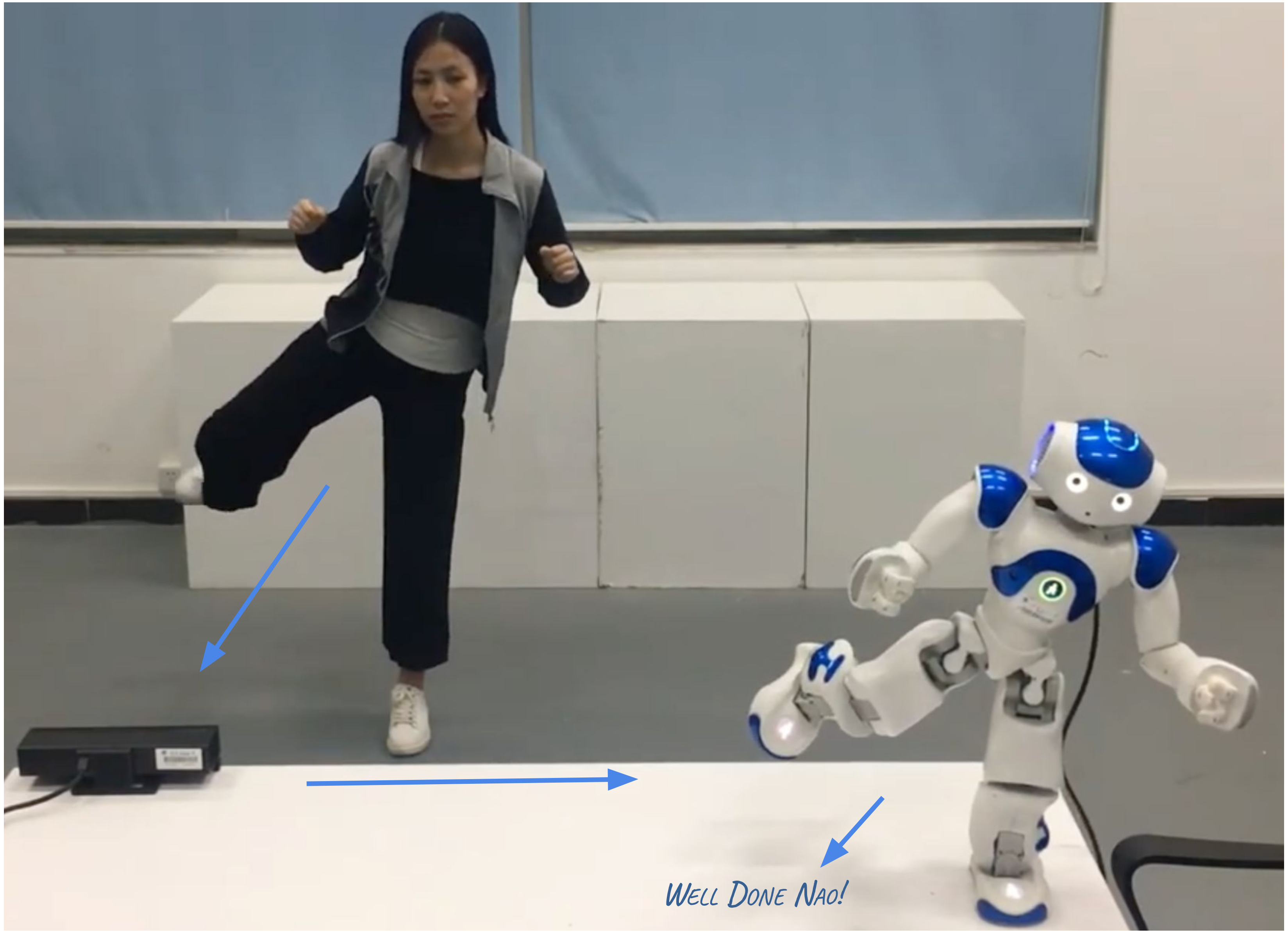
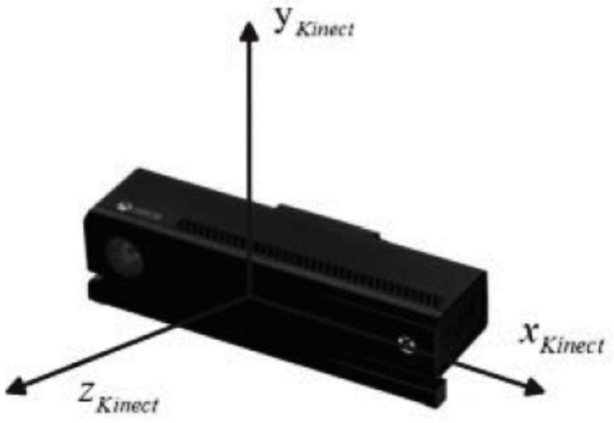
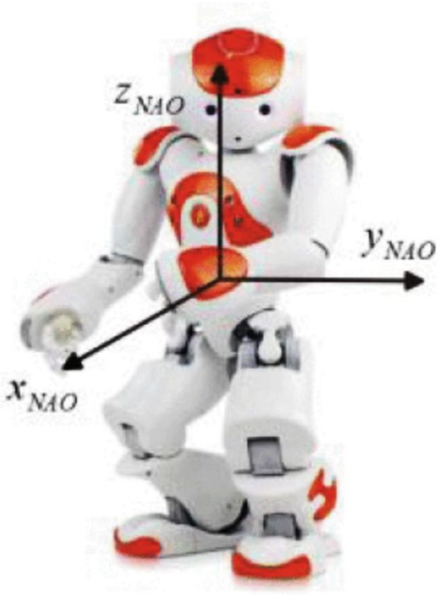


Image source:

System

- **Kinect** is a low cost **RGBD** sensor with 3 lenses:
  - RGB color camera
  - 3D optical depth sensor with infrared transmitter
  - Infrared CMOS camera for acquiring depth data





- **NAO** humanoid robot is 60 cm tall, with:
  - **Cameras**, speakers, microphones
  - Prehensile hands
  - Overall **25** degrees of freedom (**DOFs**)

$$\begin{bmatrix} x_{\text{NAO}} \\ y_{\text{NAO}} \\ z_{\text{NAO}} \end{bmatrix} = \begin{bmatrix} -z_{\text{Kinect}} \\ -x_{\text{Kinect}} \\ y_{\text{Kinect}} \end{bmatrix}$$

Data Processing

- Kinect – NAO's **coordinate system transformation**

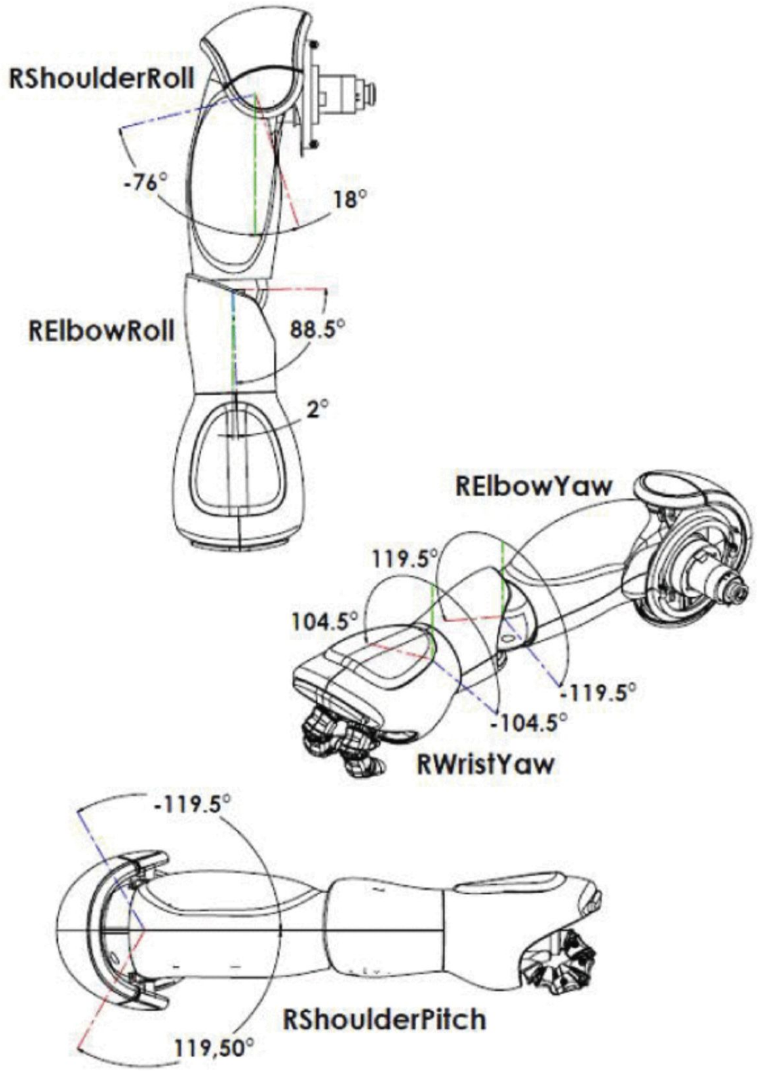
Images source: Paper*

- Three-axis vector torso coordinate system:

$$\begin{bmatrix} x_{torso} \\ y_{torso} \\ z_{torso} \end{bmatrix} = \begin{bmatrix} \cos x_x, \cos y_x, \cos z_x \\ \cos x_y, \cos y_y, \cos z_y \\ \cos x_z, \cos y_z, \cos z_z \end{bmatrix}$$

- Example, **right arm** :
  - Coordinate origin in the **shoulder** :

$$P'_{\mathrm{RS}} = O$$
$$P'_{\mathrm{RE}} = R_{\mathrm{torso}}\left(P_{\mathrm{RE}} - P_{\mathrm{RS}}\right)$$
$$P'_{\mathrm{RW}} = R_{\mathrm{torso}}\left(P_{\mathrm{RW}} - P_{\mathrm{RS}}\right)$$



Images source: Paper*

SCHOOL FOR ADVANCED STUDIES LUCCA

UNIVERSITÀ DEGLI STUDI FIRENZE

- 4 joint angles in right arm:

$$\theta_{\mathrm{RSP}} = \arctan \frac{z'_{\mathrm{RE}} - z_{\mathrm{RS}}}{x'_{\mathrm{RE}} - x'_{\mathrm{RS}}}$$

$$\theta_{\mathrm{RSR}} = \arctan \frac{y'_{\mathrm{RE}} - y'_{\mathrm{RS}}}{\sqrt{\left(x'_{\mathrm{RE}} - x'_{\mathrm{RS}}\right)^2 + \left(z'_{\mathrm{RE}} - z'_{\mathrm{RS}}\right)^2}}$$

- Moving origin to the **right elbow** joint and rotate coordinate system around y-axis and z-axis
- Then :

$$P''_{\mathrm{RE}} = O$$

$$P''_{\mathrm{RW}} = R_z\left(\theta_{\mathrm{RSR}}\right) R_y\left(\theta_{\mathrm{RSP}}\right)\left(P'_{\mathrm{RW}} - P'_{\mathrm{RE}}\right)$$

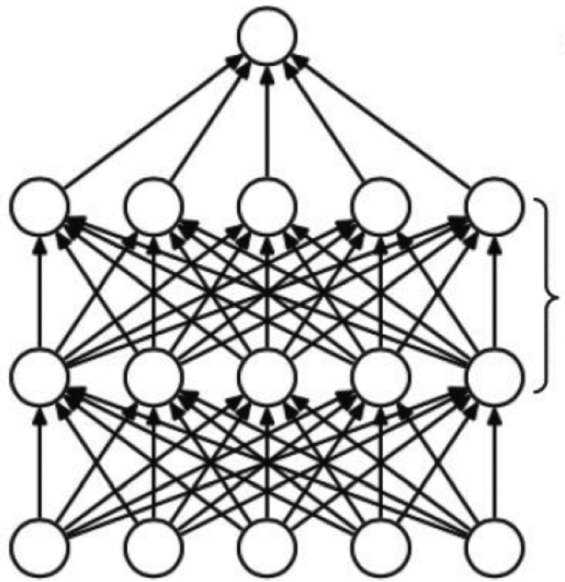
$$\theta_{\mathrm{REY}} = \arctan \frac{z''_{\mathrm{RW}} - z''_{\mathrm{RE}}}{y''_{\mathrm{RW}} - y''_{\mathrm{RE}}}$$

$$\theta_{\mathrm{RSR}} = \arctan \frac{\sqrt{\left(y''_{\mathrm{RW}} - y''_{\mathrm{RE}}\right)^2 + \left(z''_{\mathrm{RW}} - z''_{\mathrm{RE}}\right)^2}}{x''_{\mathrm{RW}} - x''_{\mathrm{RE}}}$$

Upper limb			
Left		Right	
joints	DOFs	joints	DOFs
Shoulder (P_{LS})	LSP	Shoulder (P_{RS})	RSP
	LSR		RSR
Elbow (P_{LE})	LEY	Elbow (P_{RE})	REY
	LER		RER
Wrist (P_{LW})	not used	Wrist (P_{RW})	not used
Lower Limb			
Left		Right	
joints	DOFs	joints	DOFs
Hip (P_{LH})	LHP	Hip (P_{RH})	RHP
	LHR		RHR
Knee (P_{LK})	LKP	Knee (P_{RK})	RKP
Ankle (P_{LA})	LAP	Ankle (P_{RA})	RAP
	LAR		RAR

Images source: Paper*

- MLP is a fully connected neural network, a deep learning structure. Made by:
    - Input layer – robot **joint angles**
    - Hidden layers
    - Output layer – robot **stability**
- Pre-trained supervised machine learning algorithm:
    - **Expected outputs** $\hat{Y}^k = \left(y_1^k, y_2^k, \cdots, y_n^k\right)$
    - **Real** $Y^k = \left(y_1, y_2, \cdots, y_n\right)$
- **Loss** function: $L(W,B) = -\frac{1}{n}\sum_{i=1}^{n}\hat{Y}_i \log Y_i$

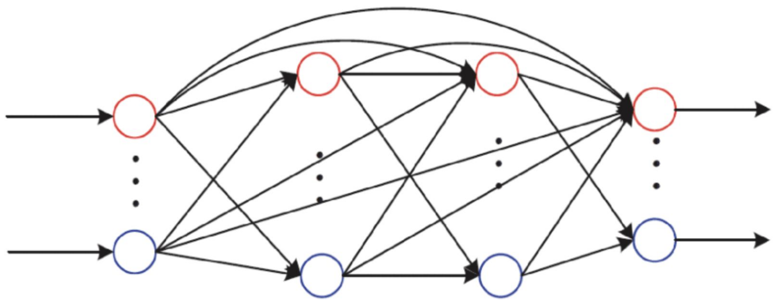


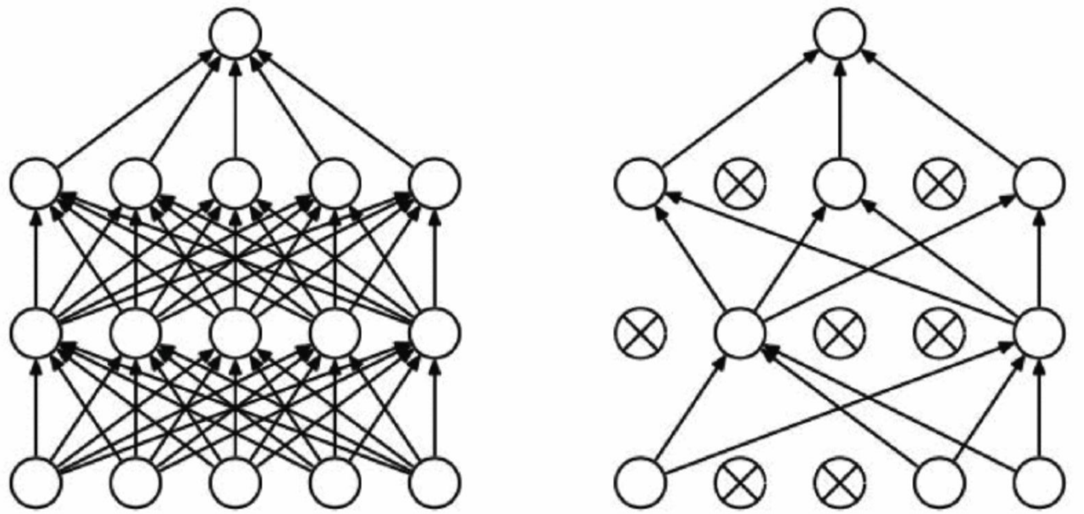
- **NN is updated by Gradient Descent** method:
    - W, $w_{ji}^l = w_{ji}^l - \eta\left(\frac{\partial L}{\partial w_{ji}^l}\right)$ weight of i-th node of layer $l-1$ to j-th node of layer l
    - B, $b_j^l = b_j^l - \eta\left(\frac{\partial L}{\partial b_j^l}\right)$ bias of j-th node of layer l
        - η – neural network **learning rate**

Images source: Paper*

- NN has been improved with **dense connection** , in order **to characterize more features** :
  - Node inputs include **not only upper layer outputs, but also previous ones**
  - Accuracy: Trad 95.5% → Optimized 96.2%
- During training **model often overfit training data** , showing **poor generalization** performance

  To prevent it:
  - **Dropout** : temporarily discard some NN units from network **according to a certain probability** . For N-nodes NN, apply dropout can collect 2N models, with a constant number of parameters to be trained, **improving over-fitting problem** and not increasing computational burden





Images source: Paper*

- **Regularization** : add a term to the original **loss** function, like L2 term (Ridge):

$$L = L_0 + \frac{\lambda}{2n}\sum_w w^2$$

  - **λ regular coefficient** , **weighing the proportion** of regularization and L0

- The **weight update** formula:

$$w = \left(1 - \frac{\eta\lambda}{n}\right) + \eta\frac{\partial L_0}{\partial w_0}$$

  - η, λ and n are positive makin' w less than 1, which is called weight decay
  - The smaller the weight w, the **less complex the network** , thus the **less likely it is to over-fitting**

Images source: Paper*

- **1500 sets** of randomly selected stable and unstable data **to train** the MLP neural network
- **500 sets** of data are collected **to test** the accuracy of the classification of neural network
- **Output** data observing if the robot is **stable** or not
- **Flow chart** shows the **imitation process**, including stability classification

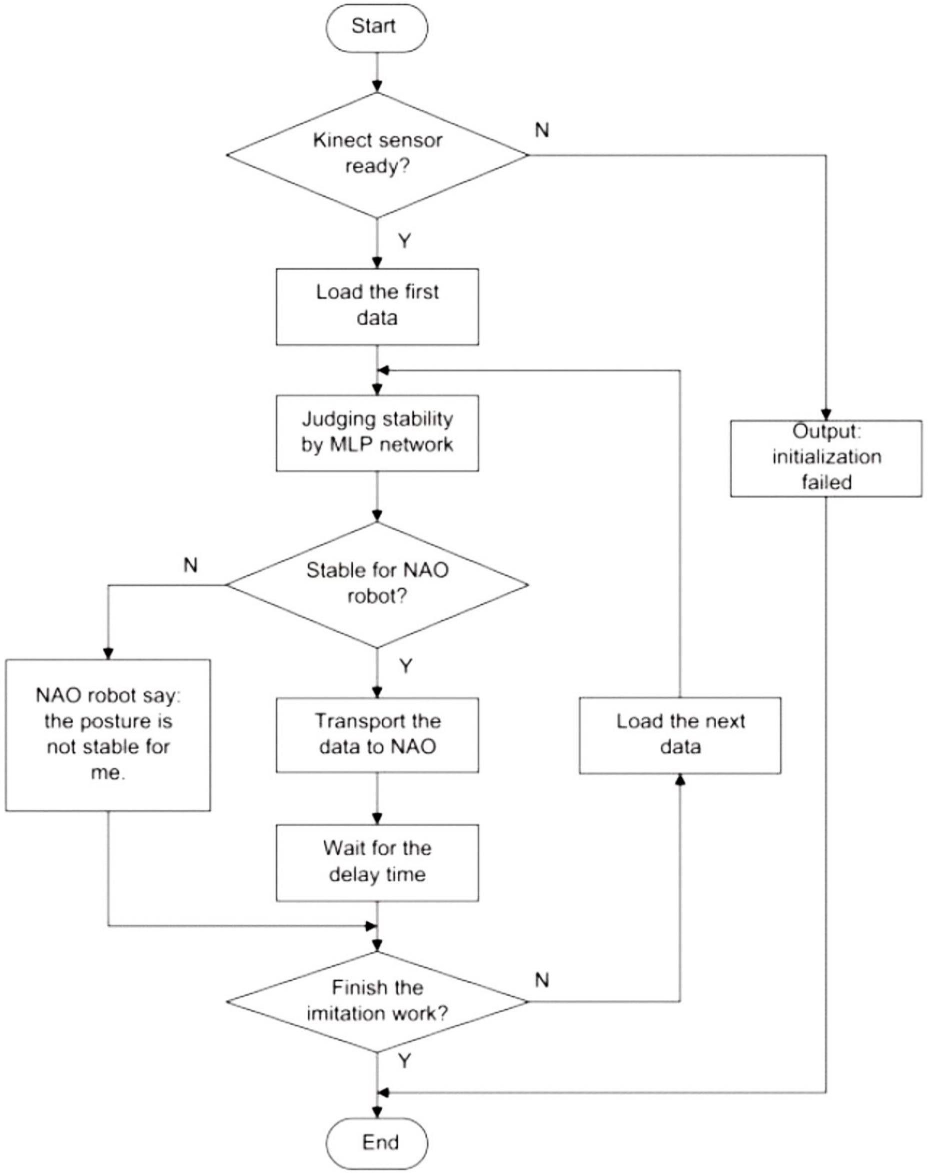
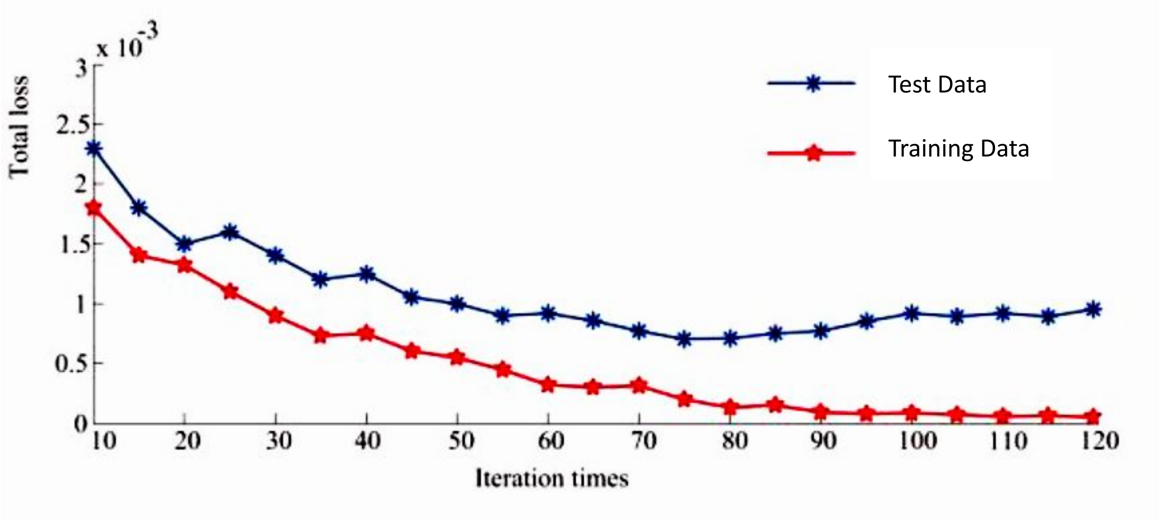


Image source: Paper*

- NN parameters adjusted to make test **accuracy high** :
  - Number of hidden **nodes**
  - Number of **iterations**
  - **Learning rate**

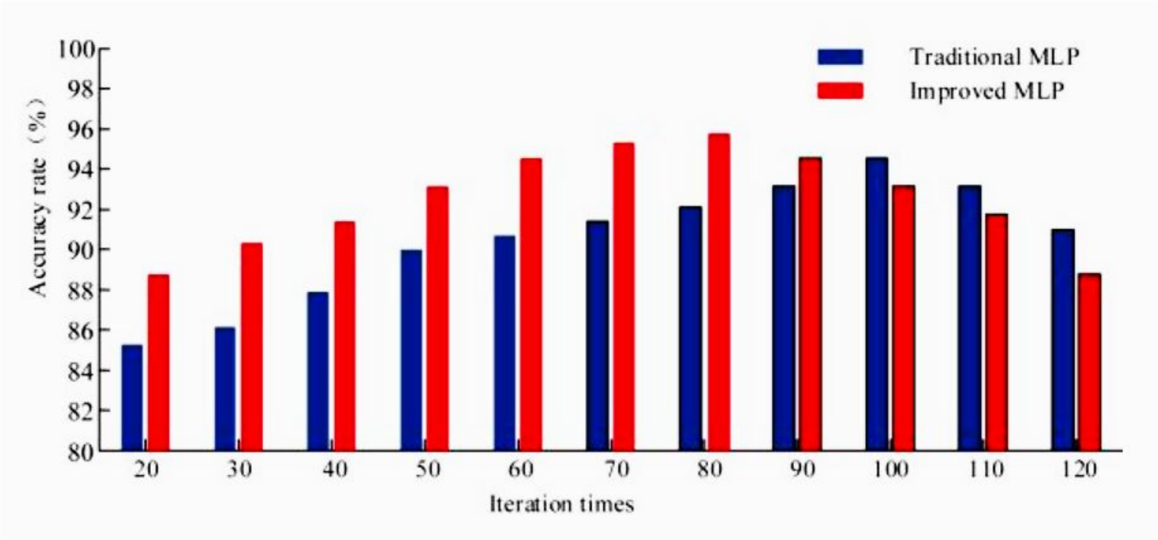
- Best **performance over test** accuracy with:
  - **16** and **8 nodes**, in hidden layers 1 and 2
  - Optimal model accuracy at **iteration times 75**

Layer1	Layer2	Training data	Testing data
4	4	90.5	86.8
8	4	92.6	89.6
8	8	94.2	93.8
16	8	96.8	95.6
16	16	97.2	95.2
32	16	97.7	94.4
32	32	98.3	90.6



Images source: Paper*

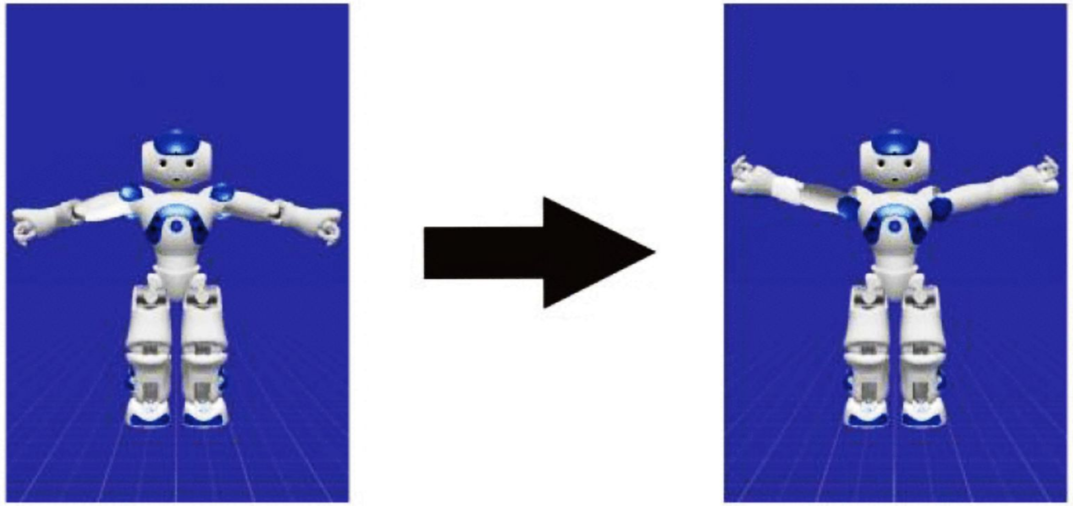
- Accuracy rate of **traditional** MLP and **improved** MLP on testing data



- Adding **dense connection** , **regularization** and **dropout** , the highest performance with an **accuracy up to 96.2%**

Image source: Paper*

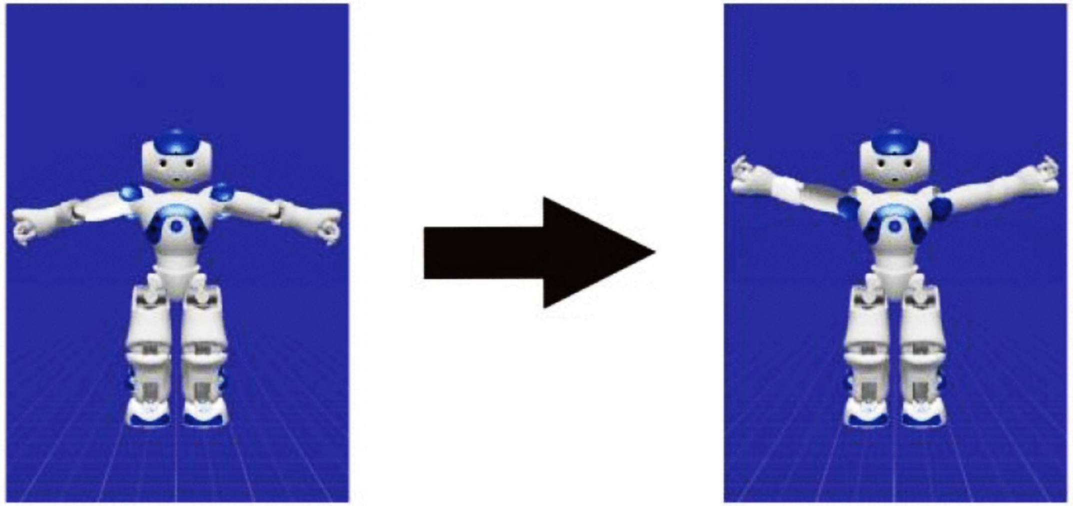
- **Robot vs human body difference** and Kinect's own noise may **cause angle mutation**, affecting the stability and accuracy of imitation experiment
- For example, when the person **moves his arms from bottom to top**, a simple arms-shoulders action, the **robot shoulder motor is required to rotate 180°**
- In addition, each **articulated motor** of NAO robot has its own **rotation limit**. If angle obtained by data conversion exceeds this limit, the robot structure will be impacted. So, robot joint motor angle is set by the robot's maximum rotation angle



1. If current angle **value exceeds maximum**, this maximum angle value is used as output to drive joint motor
2. If current angle **value is less than minimum**, this minimum angle value is used as output to drive joint motor
3. Otherwise, output according to the **actual calculation of the angle**

Image source: Paper*

- **Sliding average filtering method** : to overcome accuracy problem caused by structural differences and noise
  1. Take some consecutive samples values as a **queue**
  2. Each time put a new data into the queue and discard the first entered | **First In, First Out**
  3. Calculate queue data **arithmetic average** in the queue



- Adding the **limit** and moving **average filter**, motions become **smooth** and prevent the speed pulse
- When speed is slow, filters do not have influence on original data

Algorithm	MLP	RBF	SVM	ELM
Accuracy (%)	96.2	93.2	92.6	81.5

- More machine learning algorithms, such as **RBF** neural network, **SVM** and **ELM** have been used
- These are compared with **MLP** neural network, which can obtain the **best identification accuracy**

Images source: Paper*

SCHOOL FOR ADVANCED STUDIES LUCCA

UNIVERSITÀ DEGLI STUDI FIRENZE

Outcomes

- Some problems existing in robot imitation: **smoothness, space-time consistency and safety** are achieved
- The proposed **approach** utilizes features of neural network that is **easy to model**
- With **MLP neural network**, the stability **identification rate** is **high**
- Implemented **filters** can **overcome accidental large angle variation** to some extent

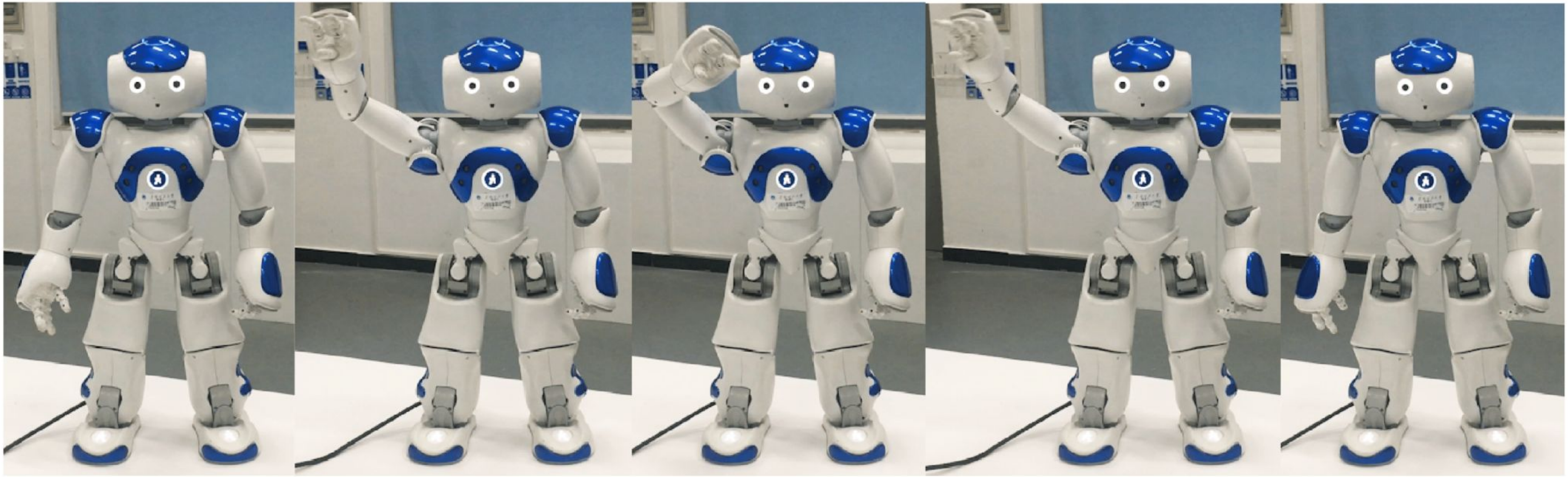
Improvements

- There are still some shortcomings need to be solved and improved:
  - e.g. the **stability under high-speed** operation is still hard to identify.

    The problem of robot stability under the **influence of inertia may be considered**

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

UNIVERSITÀ DEGLI STUDI FIRENZE



Thank you

Image source:
https://www.mdpi.com/2076-3417/8/10/2005

Images and text have been gathered from the paper* “ *Pose Analysis of Humanoid Robot Imitation Process Based on Improved MLP Network*”;
Shuhuan Wen, Yang Liu, Leibo Zheng, Fuchun Sun and Bin Fang; Proceedings of the 2nd WRC Symposium on Advanced Robotics and Automation 2019; Beijing,