Journal of Intelligent & Robotic Systems (2022) 104:59
https://doi.org/10.1007/s10846-021-01553-5

SHORT PAPER

Two-Legged Robot Motion Control With Recurrent Neural Networks

Bahadır Çatalbaş [1] · Ömer Morgül [1]

Received: 6 August 2020 / Accepted: 3 December 2021

Abstract
Legged locomotion is a desirable ability for robotic systems thanks to its agile mobility and wide range of motions that it provides. In this paper, the use of neural network-based nonlinear controller structures which consist of recurrent and feedforward layers have been examined in the dynamically stable walking problem of two-legged robots. In detail, hybrid neural controllers, in which long short-term memory type of neuron models employed at recurrent layers, are utilized in the feedback and feedforward paths. To train these neural networks, supervised learning data sets are created by using a biped robot platform which is controlled by a central pattern generator. Then, the ability of the neural networks to perform stable gait by controlling the robot platform is examined under various ground conditions in the simulation environment. After that, the stable walking generation capacity of the neural networks and the central pattern generators are compared with each other. It is shown that the inclusion of recurrent layer provides smooth transition and control between stance and flight motion phases and L_2 regularization is beneficial for walking performance. Finally, the proposed hybrid neural network models are found to be more successful gait controllers than the central pattern generator, which is employed to generate data sets used in training.

Keywords Robot locomotion control · Biped robot · Recurrent neural networks · Long short-term memory · Central pattern generator

1 Introduction

Nowadays, robotic solutions have gained huge popularity due to the recent advancements in artificial intelligence and control theory. For this reason, different robotic systems are being developed for various purposes in the industry and military. Legged robots are an interesting sub-branch of robotics with their differences and opportunities. Humanoid robot Atlas showed that legged robots have capability of agile mobility and wide range of motion with their design close to the humans [1, 2]. Unfortunately, their hybrid dynamic structure, which is inevitable for legged locomotion, includes nonlinear components that make it difficult to model and control these platforms, [3]. In the control of single-legged robots, deadbeat type controllers can be applied by dividing the motion into phases, see e.g. [4]. Moreover, different control theory techniques such as proportional-derivative (PD) and model predictive controllers can be used in the control of biped robots, see e.g. [5]. In addition, different dynamic models have been developed for the control and identification of legged robots with non-linear structures such as the Spring-Loaded Inverted Pendulum model, see e.g. [6, 7].

Stability of walking behavior is a very complex problem for legged robots due to their highly nonlinear models. Various methods are available in the literature for the analysis of the stability of walking behavior. Among these, Center of Gravity (COG) method requires that the center of mass of the legged robot be above the support polygon for the stability of walking behavior, [8]. In the Zero Moment Point (ZMP) approach, stability is achieved when the projection of point where sum of active forces is equal to zero is over the support polygon, [9]. In addition, Foot Rotation Indicator (FRI) determines stability and level of instability depending on the location of the rotation forces acting on support foot at the ground, [10]. A more advanced technique, Centroidal Moment Pivot (CMP) method eliminates the single

✉ Bahadır Çatalbaş
cbahadir@ee.bilkent.edu.tr

Ömer Morgül
morgul@ee.bilkent.edu.tr

[1] Department of Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey

Published online: 18 March 2022

foot limitation of FRI by analyzing stability depending on the location of the pivot point which is defined as the point where a parallel line to reaction force passing through the center of mass of the robot intersects with the ground, [11]. Unfortunately, these approaches become more complex when dynamically stable walking with increasing number of robot limbs is desired compared to statically stable motion.

There is also diversity in actuation techniques which affect the whole design of the legged robot. As an illustration of these, Ankaralı and Saranlı [12] introduce energy regulation method via hip torque actuation on one-legged spring-mass hopper robot. Kerimoğlu et al. [13] utilize series-elastic ankle actuation to the compass gait model and analyzed the stability of walking behavior for the resulting system. Spröwitz et al. [14] propose an open-loop motion control for a quadruped Cheetah robot by utilizing appropriate knee and hip joint actuation. These varieties of actuation also require different controller design methods. In addition to the classical controllers, there are also different legged locomotion control approaches such as central pattern generators and neural networks.

The inherent nonlinearity and the resulting complexity makes it difficult to utilize exact analytic methods for the analysis and control of legged robots. As a remedy to this problem, Sproewitz et al. [15] proposed a central pattern generator (CPG) based controller implementation in multi legged robotic systems. Likewise, Crespi and Ijspeert [16] presented a CPG based control method for a snake robot with high degree of freedom. CPGs, which mimic the reflex spring in the spinal cord, offer a control approach that does not rely on exact solutions and approximation based methods for gait control of legged robots [17–21]. CPG parameter tuning is generally performed with manual methods which can be thought of as a supervised learning procedure. During this process continuity of oscillation and phase difference between joint angles are carefully set by designer. For this reason, the tuning procedure may become difficult and prolonged with the increasing number of limbs. In addition to supervised methods, Ijspeert and Kodjabachian [22], Nakamura et al. [23], showed that non-supervised learning techniques such as evolutionary algorithms and reinforcement learning can be used to find the parameters of CPGs. There are various oscillatory models which could be utilized in the construction of CPG models, see e.g. [16]. Such a well-known model is given by Matsuoka oscillator, which is employed to generate rhythmic patterns and output of oscillator can be easily modified with tonic inputs, [24]. Due to these and some other properties, it is preferred in many CPG controlled biped robot systems, see e.g. [25–28].

CPGs need limited parameter space to create stable walking in high degree of freedom robots. Unfortunately, this requirement limits the possible range of motions of these systems to periodic trajectories. On the other hand, due to their structures, CPGs may require manual fine tuning of its parameters. In addition to this, it may not be possible to determine stable movement range due to the inclusion of the robot plant dynamic equations into the equations of CPG. Various neural network-based controllers (NNBCs) have also been developed to avoid some of these disadvantages, see e.g. [29].

Neural networks (NNs) which have found wide application areas today as an important field of the machine learning theory are inspired by biological neuron cells, see e.g. [30]. Basically, NNs consist of interconnected simplified neural cell models which are building blocks of the network. There is a wide range of neuron cell models from as simple as perceptron to complex ones such as leaky integrator and long short-term memory (LSTM), [31–33]. Some of the NN architectures can perform better in different problems. Feedforward NNs are successfully applied to classification and object detection type of problems, see e.g. [34–36]. In detail, Krizhevsky et al. [34], Redmon and Farhadi [36] showed successful utilization of convolutional neural networks (CNNs) in object classification and detection problems, respectively. There are successful implementations of recurrent neural networks (RNNs) in natural language processing and next character or word prediction in the text problems, see e.g. [37, 38]. At the same time, RNN based motion controllers can provide biped robot gait control solutions, which do not have the disadvantages of CPG such as limited motion range and manual tuning requirement, in cases where other analytical-based exact and approximated solutions are not desired or cannot be applied, see e.g. [39, 40]. Unlike the CPG parameter tuning case, there are well-defined methods to adjust the parameters of NNs, which are generally called as learning algorithms. Various learning algorithms, such as supervised, semi-supervised or unsupervised, could be utilized in the training of NNs, see [31, 41]. Among these, supervised learning algorithms are frequently utilized to train NNs when input and output data sets are available, see e.g. [30]. To achieve this, different optimizers are developed and utilized such as Adam and SinAdaMax, see e.g. [42, 43].

Nowadays, spiking neuron models which are employed in neuromorphic engineering studies are successfully applied to robot control problems to generate CPGs and NNs. Rostro-Gonzalez et al. [44] proposed a CPG system based on spiking neurons which are trained with Simplex method for hexapod robot locomotion such as walking, jogging, and running gait types. This work focused on the generation of digital hardware-compatible locomotion controllers with high computational efficiency. In the proposed controller, a different connection topology is required to change gait type and it was performed by changing synaptic weight matrices. Jaramillo-Avila et. al. [45] benefitted from a spiking neural network (SNN) to process sensorial information

from address event representation-based camera images to decide movement direction of biped robot. In [45], weights of the proposed SNN were found experimentally. Guerra-Hernandez et. al. [46] performed real-life experiments of the CPG system based on spiking neurons in biped, quadruped, and hexapod robots. They benefitted from grammatical evolution and Victor-Purpura distance-based fitness function to tune connection weights. In this controller, the gait transition can be performed by resetting all neuron membrane potentials and then setting initial states of spike trains for desired gait type. Gutierrez-Galan et al. [47] proposed a combination of three spiking central pattern generators (SCPGs) to realize online gait type changes with reasonable transition time by using the limited number of spiking neurons. In the proposed structure, while selected SCPG begins to generate spike train, previous SCPG is inhibited, quickly. By using the developed controller, locomotion of an arthropod-like robot was successfully controlled in real-time. Even though each SCPG has a low number of neurons, the combined system requires high redundancy to be able to change between gait types, and also neuron weights were manually adjusted in this study. In addition, likewise aforementioned SCPG based locomotion controllers, sensorial information was not taken into account. In these successful studies, the spiking neuron model is utilized to control locomotion at different abstraction levels. Generally, sensorial information, which can be beneficial especially in inclined or rough terrain, was not processed in SCPG-based locomotion controllers. This design selection may be related to the limited processing or learning capability of the spiking neuron model compared to LSTM type recurrent neuron models and the necessity of parameter tuning to extract information from sensorial inputs. Recurrent neural networks have well-defined learning techniques and they may accomplish switching between different gait types by using the same network weights via suitable training set pattern selections. For more information on CPGs and their applications, the reader is referred to e.g. [18, 48], and the references therein.

Researchers seek optimum combinations of different control techniques and learning algorithms to benefit from the powerful sides of each of them in the legged locomotion control problem. Auddy et. al. [49] suggest a novel hierarchical control mechanism consist of a CPG and a feedforward neural network (FNN) as low and high-level controllers for bipedal locomotion, respectively. In [49], CPG was modulated by FNN to correct walking direction by using only two connections. In detail, a combination of manual tuning and genetic algorithm was utilized to find parameters of CPG, first. After basic walking behavior acquired via tuned low-level controller, parameters of FNN are found by using deep reinforcement learning via further walking simulation experiments. In this control scheme, lateral deviations from straight direction were corrected with two gains, which were the outputs of FNN, multiplied by sagittal hip oscillator CPG outputs. Mandava and Vundavilli [50] benefit from a newly proposed modified chaotic invasive weed optimization (MCIWO) and well-known particle swarm optimization (PSO) algorithms to find weights of a feedforward neural network that is responsible for adaptively tuning gains of torque-based proportional-integral-derivative (PID) controller during the bipedal locomotion. First, ZMP and inverse kinematics concepts were utilized to find dynamically balanced walking gaits. Later, neural networks are trained with MCIWO, PSO, and the traditional steepest descent algorithm to estimate the best PID gains to track walking gaits. After that, the performances of neural networks are tested in the simulation environment. Finally, selected neural network weights were employed to estimate PID gains of a real biped robot which was walking on ascending and descend ing slope surface between $[-5, 5]$ degree interval in this study. To sum up, multiple parameter tuning stages needed to be followed in these recent studies because the proposed controllers consisted of a combination of different types of systems such as PID, CPG, and FNN. Unfortunately, these design selections may bring their own disadvantages such as controller design complexity and training difficulty. One reason for the requirement of controller combination may be related to the absence of the memory property of employed feedforward neural networks. On the other hand, the LSTM neuron model has the ability to solve long-term relationships with its memory property and it is resistant to vanishing gradient problems. For these reasons, LSTM recurrent layer has some advantages compared to feedforward layers which are utilized in these two studies, and its usage may enhance the abilities of proposed controllers in [49] and [50].

In a similar way, well-known Wiener and Hammerstein system identification models employ a combination of a feedforward neural network and polynomial to represent plant dynamics, see e.g. [51]. Since there is no memory term in feedforward neural networks, polynomial block accompanies feedforward neural network block to add memory property to system identification scheme in a similar manner to CPG in the hierarchical control mechanisms and integrator term in PID controllers. From this perspective, it is seen that feedforward neural network blocks found usage in both control and system identification literature. In our previous study, we compared the performance and parameter efficiency of feedforward and recurrent neural networks for a CPG-controlled biped robot system identification problem in an end-to-end manner, [52]. Here, the series-parallel system identification scheme was used when comparing neural networks of various depths to facilitate memory requirement and provide a fair comparison. As a result, the recurrent neural network with one regression layer and one LSTM layer reached the lowest system identification error among all network architectures. In this way, the potential of LSTM

layers has been investigated in the legged locomotion identification problem and it is found that their end-to-end usage is possible and efficient for bipedal locomotion identification problem. The results given in [52] may also make sense for the legged locomotion control problem because legged locomotion includes hybrid dynamics so tracking the walking phase changes is required for high-performance locomotion control. For this reason, the usage of recurrent neural networks with LSTM layers needs to be investigated in the biped locomotion control problem, extensively.

In this study, we propose the utilization of neural network-based controllers which consist of feedforward and recurrent neuron layers as torque and position controllers for biped robot locomotion. Compared to CPG type controller, proposed NNBC can behave like a meta learner. In detail, it may generalize input and output relations of the CPG controllers under suitable training conditions. We note that in this work we utilized CPGs to generate joint angle and torque data which we used to train the NNBCs in achieving successful walking or biped robots. Naturally if similar data could be generated or obtained by other means, our approach could be utilized by using such data as well. Simulation or computer animation tools are natural candidates to generate such data and recently their usage in robotics and related areas become an important research area [53]. Such an approach could be utilized to generate appropriate joint angle or torque values [54]. Since we focus on CPG based data, in this work we do not pursue such an approach. It should be noted that utilization of such data, along with data obtained by CPG's, is an interesting research problem that deserves further investigation. The literature is rich on this subject, and for further information see e.g. the following works and the references therein [55–57].

Analytical approaches are frequently employed to sustain stable legged locomotion with classical control techniques in the literature [48]. First, gait type is determined by analytical methods or observation in the path planning stage. Then joint trajectories are derived by benefiting from inverse kinematics equations under selected stability constraints such as COG, ZMP, FRI, CMP, etc. Later on, required joint torques may be calculated by inverse dynamic equations, if possible, or classical control techniques such as PID see e.g. [58].

Unfortunately, legged robot dynamics include highly nonlinear terms and they show hybrid dynamic behavior due to the nature of walking so the resulting equations contain complex expressions. For these reasons, various simplification methods are proposed to ease the analysis of the robot model and the application of classical control techniques for controlling legged locomotion in the literature. One of them is applied by ignoring extremities with low weight such as legs in the robot model see e.g. [58–61]. In this way, analysis of the robot model gets easier while the model accuracy decreases. However, the performance of some control methods may depend on the accuracy of the utilized robot model. In other words, inaccuracies due to simplification may decrease control performance or cause unstable behavior depending on the level of simplification [48]. For instance, ignoring limb weight to ease COM calculations may violate stability criteria such as reported in the [59].

On the other hand, the limited parameter space of classical controllers may not perform well for varying robot dynamic properties or terrain conditions so various model predictive control (MPC) architectures are proposed in the literature for biped locomotion, see e.g. [61]. In classical control techniques such as PID, there is a limited number of controller parameters that can be tuned to track trajectories for low-level control scenarios. Even though MPC architectures relax this limitation, they may require complex controller designs and an external gait generator block is added into the control diagram to drive low-level controller. On the other hand, recurrent neural networks can be utilized as controllers for both low-level and high-level control problems at the same time with their large tuneable parameter space like in their biological counterparts. Moreover, proposed NNBCs in this study are trained by using the original robot model dynamics with all nonlinear terms. This situation may increase harmony between the controller and the robot model. Beyond these advantages of the proposed NNBCs compared to classical control approaches, utilization of LSTM layers in the neural controllers such as in our proposed controller may make designing high-performance neural model reference control architectures possible.

As the main contribution of this work, we propose NNBCs which involve feedforward and recurrent layers rather than classical feedforward neural network-based controllers for biped robot locomotion control. Since the biped robot model shows hybrid dynamic behavior due to changes between flight and stance phases for each leg, we expect that the recurrent layer which consists of LSTM type neurons may contribute to tracking these changes and control locomotion successfully. As a second contribution, to support this idea the proposed hybrid neural controllers are utilized in various combinations where they are placed in the feedback loop and feedforward paths. Moreover, these are also employed together with PID controller for gait control. As a third contribution, we proposed two NNBCs which generate position and torque outputs. By using these neural controllers we have performed various simulations under varying ground conditions. The results of these simulations are analyzed within the scope of this paper to demonstrate the superiority of proposed NNBCs against its CPG and PID type controller alternatives. As a final contribution, generalization abilities of proposed NNBCs are demonstrated and the effects of hyperparameter selection such as network size, mini-batch size and L_2 regularization are reported in the generalization performance depending on the placement of controller. Our results indicate that the proposed NNBCs

perform better in many cases for a wide range of ramp angle, walking speed and rough terrain environment than their CPG and PID based counterparts.

2 Problem Definition

This paper focuses on biped robot locomotion control by using NNBCs which consist of feedforward and recurrent layers. In this section, we present biped robot model, CPG which is employed to generate data sets and basics of hybrid neural network architecture which will be utilized in control of two-legged robot.

2.1 Biped Robot Model

The two-legged robot used in this study consists of a point mass hip, 4 rigid limbs and 6 joints, as shown in Fig. 1 and its equations of motion, which are given in the Appendix 1 for the sake of completion can be obtained by using standard methods of mechanics, see e.g. [27] for further information. Moreover, horizontal and vertical friction and ground reaction forces are also modeled with parallel spring and damper between the ground-contacting foot and the ground as seen in Fig. 1.

The generalized coordinate vector of the biped robot system can be found in a similar way to [62, 63] and it is represented with the 6 degrees of freedom as shown below:

$$\boldsymbol{q} = [x_1, x_2, \theta_1, \theta_2, \theta_3, \theta_4]^T. \quad (1)$$

The robot model is controlled with the torques created in the six joints which are at hip, knee, and ankle joints at both legs as follows:

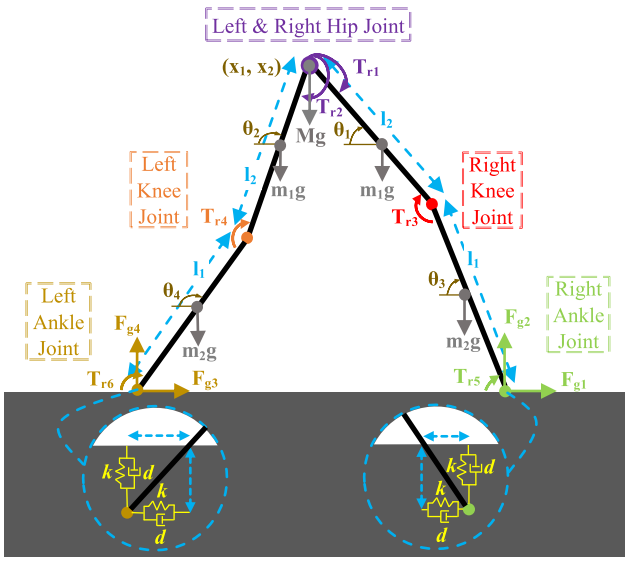


Fig. 1 Representation of biped robot model joints, limbs, torques, limb angles, and interaction with the ground

$$\boldsymbol{T} = [T_{r1}, T_{r2}, T_{r3}, T_{r4}, T_{r5}, T_{r6}]^T. \quad (2)$$

In the single support phase, ankle torque of the foot which is in the flight phase does not affect robot model dynamics so ankle torque of that foot is excluded from the torque vector in Eq. 2.

The elastic ground contact model which is defined with parallel spring and damper permits horizontal and vertical movement of the support feet such as slip over walking surface and sinking through the ground. Thus, the degree of freedom of the robot model does not decrease at single or double support phases, which is different from [62, 63] where hard ground contact model is utilized. Under these conditions, the biped model becomes a fully-actuated system at the double support phase. However, the removal of ankle torque in the flight phase makes the overall system underactuated in the single support phase. Moreover actuated ankle joint makes it possible to perform static stable walking [64]. In addition to joint torques, ground reaction and frictional forces also affect the robot movement during the stance phase of each leg.

The parameters utilized in the Fig. 1 are described in Table 1. The ramp angle between the walking plane and the ground plane is not shown here, in order not to complicate the figure.

2.2 Central Pattern Generator

Since our aim is to utilize NN structures to control the motion of biped robots, we need meaningful data representing successful walking for the training of our proposed networks. CPGs which consist of combination of rhythm generators produce periodic waveforms for joint angles and/or torques to provide stable and periodic locomotion, see e.g. [65]. Taga et al. [27] showed that coupled structure of CPG and biped robot is capable of performing stable locomotion by choosing the parameters of CPG, appropriately. Unfortunately, an analytical method to choose appropriate

Table 1 Explanation of terms used in robot model

Paramater	Description
x_1, x_2	Horizontal and vertical hip coordinates
$\theta_1, \theta_2, \theta_3, \theta_4$	Limb angles
l_1, l_2	Lower and upper leg limb lengths
m_1, m_2	Mass of upper and lower leg limbs
g	Gravitational acceleration
M	Body mass
$T_{r1}, T_{r2}, T_{r3}, T_{r4}, T_{r5}, T_{r6}$	Torque values and directions
$F_{g1}, F_{g2}, F_{g3}, F_{g4}$	Ground reaction and friction forces
k	Spring coefficient
d	Damping coefficient

parameters is not known and the whole tuning should be done manually. We utilize this CPG driven two legged robot model to obtain the data set related to successful walking, see [27]. The utilized CPG equations take feedback from the biped model which has interaction with the walking surface and calculates the torques which will be applied to robot model joints. In detail, the CPG model utilized in this work consists of one Matsuoka oscillator per joint, hence the total structure contains 6 such oscillators as demonstrated in Fig. 2. Moreover, each oscillator contains one primary and one supplementary leaky integrator neuron. Hence, CPG structure contains 12 coupled leaky integrator models as given below ($i = 1, ..., 12$):

$$\tau_i \dot{u}_i = -u_i + \sum_{i,j=1}^{12} w_{ij} y_j - \beta v_i + u_0 + Feed_i(\boldsymbol{x}, \dot{\boldsymbol{x}}), \tag{3}$$

$$\tau_i' \dot{v}_i = -v_i + y_i, \tag{4}$$

$$y_i = \max(0, u_i), \tag{5}$$

where variables u_i, w_{ij}, y_j, v_i, u_0, $Feed_i$, τ_i and τ_i' correspond to internal state of i^{th} neuron, coupling coefficient from j^{th} to i^{th} neuron, output of j^{th} neuron, self-inhibition of the i^{th} neuron, speed excitation level input, feedback from the biped platform to i^{th} neuron where $(\boldsymbol{x}, \dot{\boldsymbol{x}})$ are the biped internal states, and time constants, respectively. Robot model feedback connections are taken into CPG model with $Feed_i(\boldsymbol{x}, \dot{\boldsymbol{x}})$ function which is detailed in the Appendix 1. The outputs y_i of CPG are utilized to generate the joint torque inputs for the biped as

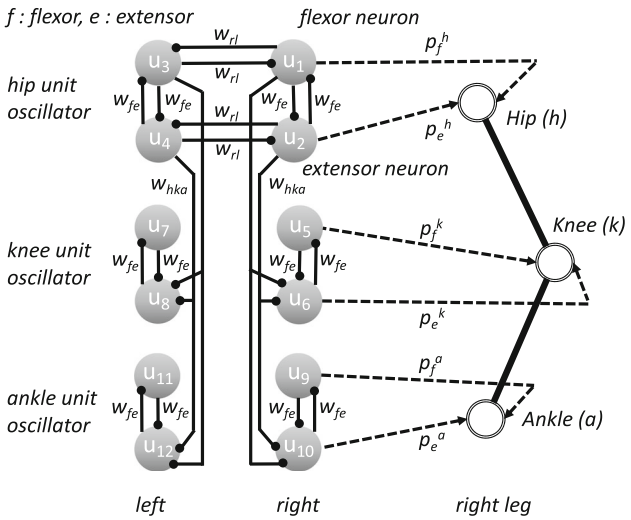


Fig. 2 Neural rhythm generator. Weights of interconnections are represented with w_{fe} w_{rl} w_{hka} and neural rhythm generator torque multipliers are shown with p_f^h, p_e^h, p_f^k, p_e^k, p_f^a, p_e^a

$$T_{r1} = p_e^h y_2 - p_f^h y_1, \; T_{r2} = p_e^h y_4 - p_f^h y_3, \tag{6}$$

$$T_{r3} = p_e^k y_6 - p_f^k y_5, \; T_{r4} = p_e^k y_8 - p_f^k y_7, \tag{7}$$

$$T_{r5} = (p_e^a y_{10} - p_f^a y_9) \max(0, F_{g2}), \tag{8}$$

$$T_{r6} = (p_e^a y_{12} - p_f^a y_{11}) \max(0, F_{g4}), \tag{9}$$

where p_f^h, p_e^h, p_f^k, p_e^k, p_f^a, and p_e^a coefficients correspond to torque multipliers which are again manually hand-tuned to generate stable locomotion. The speed excitation value u_0 is a tonic input that has an amplifying effect on the oscillation amplitude of CPG as given in Eqs. 3–5. Increase of oscillation amplitude results in higher joints torques which cause higher acceleration of limb movements as shown in Eqs. 6–9. Hence, the CPG driven robot model walking speed increases with increasing speed excitation value. For further information, see [27].

2.3 Neural Network Based Controller Design

The use of NNs as controller, which is currently a widely investigated research topic, allows the addition of nonlinear plant dynamics that are ignored or linearized because of the limited parameter space of classical controllers during the controller design process, see e.g. [66]. With this motivation, we propose hybrid neural network (HNN) structures which consist of feedforward and recurrent layers as controller for biped locomotion control in this work.

Generic form of a multi layer feedforward NN is demonstrated in Fig. 3. Typically inputs of a layer are multiplied with weights and their summation is passed through a nonlinear activation function to obtain the outputs of this layer. This process is repeated till the output is calculated.

Let us denote the input vector as $\boldsymbol{x}$ and the output vector as $\boldsymbol{o}$. Their relation may symbolically be given as:

$$\boldsymbol{o} = F(\boldsymbol{W}, \boldsymbol{x}), \tag{10}$$

where F is a nonlinear function and $\boldsymbol{W}$ symbolically represents the weights of NN. Let $(\boldsymbol{x}, \boldsymbol{d})$ be an input/output pair in our training set and $\boldsymbol{E}$ be any meaningful cost function which measures the closeness of $\boldsymbol{d}$ and $\boldsymbol{o}$. The basic problem can be recast as an optimization problem as given below:

$$\boldsymbol{W}_{opt} = \min_{\boldsymbol{W}} \boldsymbol{E}. \tag{11}$$

This optimization can be achieved by using the well-known back-propagation algorithm, see [31].

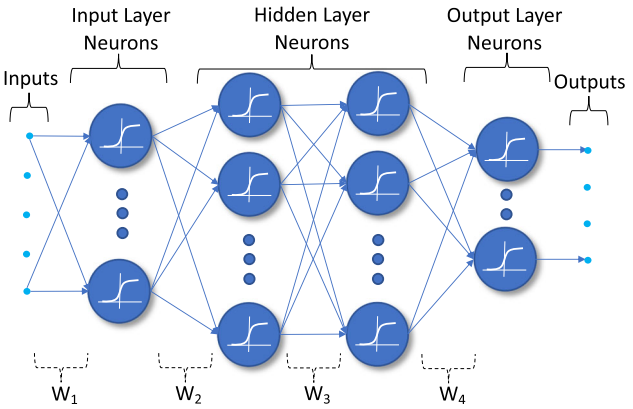


Fig. 3 Feedforward NN where W_1, W_2, W_3 and W_4 show the weight matrices of layers

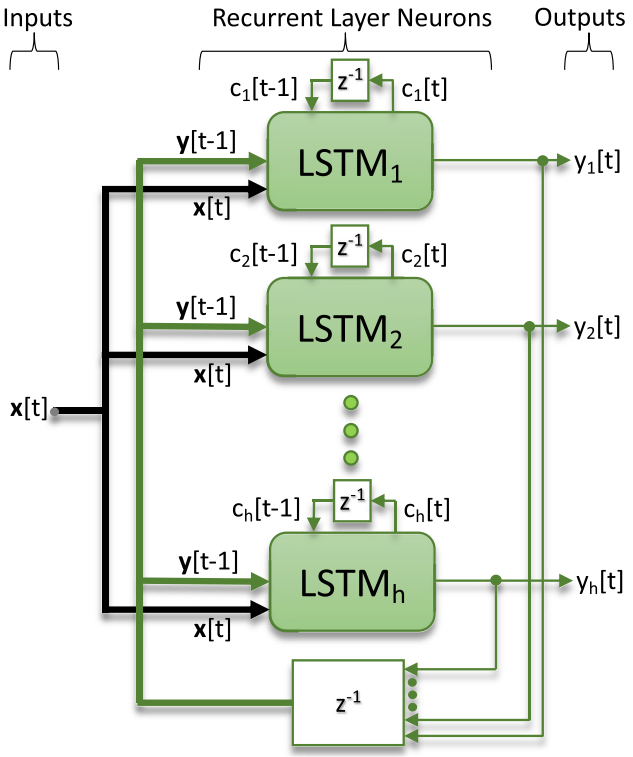


Fig. 4 Recurrent layer with LSTM neuron cells where z^{-1} denotes one step time delay

Figure 4 shows the generic structure of a LSTM recurrent layer where neurons have recurrent connections from other neurons in the same layer and also external input connections. In this case, each LSTM cell has an output $\boldsymbol{y}[t]$ as well as an internal state $\boldsymbol{c}[t]$. Symbolically, if $\boldsymbol{W}$ represents the weights of the RNN shown in Fig. 4, the input-output relation can be formally represented as:

$$\boldsymbol{y}[t] = F(\boldsymbol{W}, \boldsymbol{y}[t-1], \boldsymbol{x}[t]), \tag{12}$$

where F is a nonlinear function which depends on the activation functions utilized in the LSTM cell. Now assume that $(\boldsymbol{x}[t], \boldsymbol{d}[t])$ is an input/output sequence in the training set. The problem is to choose appropriate weights $\boldsymbol{W}$ such that when the input sequence is $\boldsymbol{x}[t]$, the output sequence $\boldsymbol{y}[t]$ is sufficiently close to the desired output sequence $\boldsymbol{d}[t]$. Similar to (11), this could also be recast as an optimization problem.

A classical way of solving the optimization problem given by (11) is to utilize the well-known gradient descent approach. In this case, the weights are symbolically updated as follows:

$$\boldsymbol{W}_{k+1} = \boldsymbol{W}_k - \alpha \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{W}_k}, \tag{13}$$

where the term $\partial \boldsymbol{E}/\partial \boldsymbol{W}_k$ is called as the error gradient, and $\alpha > 0$ is the learning coefficient. This process is typically implemented in the well-known back-propagation algorithm which consists of subsequent application of forward propagation and error back-propagation phases, see e.g. [31].

2.3.1 Forward Propagation

The general structure of NNBCs to be used in this study is presented in Fig. 5 where the combination of feedback connections from the robot platform and external reference inputs constitute input sequence $\boldsymbol{x}[t]$ which is given to the LSTM layer. Then, the output of the recurrent layer is given to the feedforward layer as input. After that, the feedforward layer performs the linear regression process by taking the weighted average of the inputs and it constitutes regression output sequence $\boldsymbol{o}[t]$ which is the joint torques and/or limb

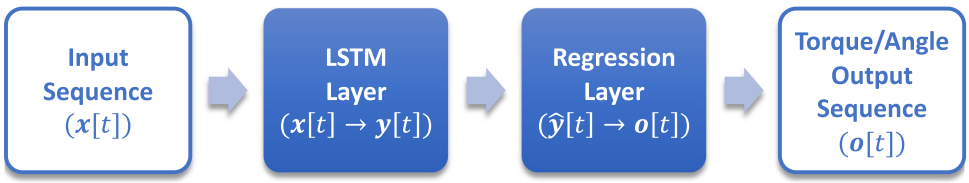


Fig. 5 Proposed HNN based controller structure. Inputs and outputs of recurrent layer (LSTM layer) are presented with $\boldsymbol{x}[t]$ and $\boldsymbol{y}[t]$. In a similar way, inputs and outputs of feedforward layer (regression layer) are shown with $\hat{\boldsymbol{y}}[t]$ and $\boldsymbol{o}[t]$

angles as the output of the NNBC depending on the control scenario. Throughout this work, NNs have been trained with $N = 1000$ steps long patterns starting at $t_0 = 0$, ending at $t_1 = 10$ seconds and sampled with $t = 0.01$ second time intervals.

LSTM neuron model has four inner gates and two states whose recurrence relations are well-known and are omitted here for brevity, see e.g. [33, 67]. The output of LSTM cell is denoted with the hidden state and it is expressed by $\boldsymbol{y}[t]$ at time step t. In our proposed network architecture the hidden state $\boldsymbol{y}[t]$ is given as input to the subsequent feedforward regression layer.

In this study, feedforward layer at output of the network is used as regression layer. Parameter matrix of the feedforward layer neurons is expressed by the variable $\boldsymbol{W}_{fnn}$. Feedforward layer inputs are chosen as expanded version of outputs of previous recurrent layer. Torque/angle output of the neural controller $\boldsymbol{o}[t]$ at the time t is calculated as:

$$\boldsymbol{o}[t] = \boldsymbol{W}_{fnn}\hat{\boldsymbol{y}}[t], \tag{14}$$

where $\hat{\boldsymbol{y}}[t] = [\boldsymbol{y}[t]\ \ 1]^T$.

2.3.2 Back-Propagation

As explained above, the weights of NN are chosen to minimize an appropriate cost function. Assume that $(\boldsymbol{x}[t]), \boldsymbol{d}[t])$ is a given input/output pattern in the training set. Then, error metric to be used in Eq. 11 is chosen as given below:

$$\boldsymbol{E} = \frac{\Delta t}{2} \sum_{t=1}^{N} (\boldsymbol{o}[t] - \boldsymbol{d}[t])^2. \tag{15}$$

Hence, to update any weight $\boldsymbol{W}$ in the NN, we need to compute the error gradients $\partial \boldsymbol{E}/\partial \boldsymbol{W}_k$, see Eq. 13. To this end, error gradient with respect to regression layer parameters is calculated with classic back-propagation under the half mean square error metric and it is used to update the parameters of $\boldsymbol{W}_{fnn}$ matrix via Adam optimizer, see [42]. After that, we utilize the well-known Back Propagation Through Time (BPTT) algorithm to compute the error gradients of the recurrent layer, see e.g. [68]. Application of this technique to find the error gradients with respect to the weights of LSTM parameters are omitted here for space limitations; they are rather straightforward and could be found in the literature, see e.g. [69].

As a final note, the cost function given by Eq. 15 is given as a standard error metric. When L_2 regularization is utilized, this cost is augmented with the term $(L_2 \boldsymbol{X}_k^T \boldsymbol{X}_k/(2\boldsymbol{n}))$ where $\boldsymbol{X}_k$ represents the weights of the network. The actual implementation of optimization will be performed by using the well-known Adam optimizer, see e.g. [42].

3 Methodology

In this section, we explain three different scenarios that are generated for evaluating stable locomotion generation performance of NNBCs. To this end, training, validation and testing data sets are generated for each of these scenarios. Then, NNs are trained to produce desired output data for the input data of patterns in the training set. Lastly, the most successful NN architecture is determined for the locomotion control problem.

3.1 Data Set Preparation

The two-legged robot driven by the CPG with the manual tuning of its parameters as in [27] is used for different speed excitation values and ramp angles to generate discrete data sets to be used in the training, validation and testing of NNs. In the scope of this work, ramp angle and speed excitation values are chosen as constants during walking simulations. Here, ramp angle value determines the slope of the walking environment and speed excitation value affects walking velocity by amplifying oscillations of CPG. Due to the limitations of CPG model, the robot model is not expected to achieve stable locomotion for each ramp angle and speed excitation value combination. Therefore, CPG-driven movements need to be classified as successful and unsuccessful gait patterns. To achieve this, CPG driven locomotion patterns are examined by an expert and motion patterns which have continuity are selected as successful locomotion patterns. Then, important metrics are determined to automate the detection of walking success such as 0.6 m/sec minimum average walking velocity, 0.6 m minimum hip height, 0.2 m maximum jumping height and ankle positions. After that, these metrics are utilized in both data set generation and NN controlled walking success evaluation.

Training, validation and testing data sets are separately produced by using different ranges of ramp angles and speed excitation values to prevent over-fitting problem. In this context, 40 uniformly distributed speed excitation values in the range of [3 10] and 31 uniformly distributed ramp angle in the range of $[-7\ \ 7]$ degrees are selected to form the training data set. The robot model is driven for 1240 different combinations of these ramp angles and speed excitation levels by CPG controller. As a result, 398 of these 1240 combinations are determined as stable walking patterns. Similarly, for the validation data set, 30 different speed excitation values in the range of [2 11] and 30 different ramp angles in the range of $[-8\ \ 8]$ degrees are run together. Before the robot model is driven for 900 different combinations of these ramp angle and speed

excitation level by CPG controller, conflicting ramp angle and speed excitation value combinations with the previous 1240 combination are eliminated from trial set. Hence, independence between data sets is preserved. Under these conditions, 195 stable walking patterns are obtained for the validation data set. Finally, in order to construct a test data set, 31 different speed excitation values in the range of [1 12] and 30 different ramp slopes in the range of [− 9 9] degrees are selected. Conflicting combinations with previous sets are eliminated again. thus, 148 stable walking patterns, which are different from the previous data sets, were obtained on the 930 configurations. According to successful locomotion criteria, CPG-driven biped robot model becomes successful at 32.1% of the training set combinations, 21.67% of the validation set combinations and 15.91% of the test set combinations. Details are summarized in Table 2.

To speed up the training of the NNBCs, which are described in the following subsections, produced data set input and output patterns are passed through decimation operation which is a signal processing technique about decreasing sample points in sequences without causing aliasing distortion. Inputs and outputs of CPG driven biped robot model which is simulated at a rate of 10 KHz for 10 seconds long are filtered with equiripple low pass filter which has 100 Hz passband cutoff frequency. Then, downsampling operation at a rate of 100 to 1 is applied to filtered data. In this way, decimation operation is completed without losing important information. After that, these locomotion data sets are re-scaled to fit the [− 0.5 0.5] range in order to facilitate the training process and allow the usage of different neuron activation functions. Thus, robot model input and output data which are re-scaled and re-sampled at 100 Hz are produced for generating various supervised learning data sets.

Within the scope of this work, the performance of NNBCs is evaluated at three scenarios. In the first scenario, joint torque generation capability of the neural controller is evaluated in a closed loop system where the neural controller takes feedback connections from the robot model and speed excitation input directly. In the second scenario, reference limb angle producing ability of neural controller structure is examined with respect to ramp angle and speed excitation value inputs. In this scenario proposed limb angles are tracked with a PID controller. In the third scenario, neural controller replacement of PID controller is performed and joint torque generation capability of the neural controller is evaluated by taking reference and actual limb angles. To this end, three different input-output data set have been produced from the re-scaled and re-sampled locomotion data sets as detailed in the following subsections.

Table 2 Data set generation with central pattern generator driven biped robot platform

	Excitation Value Interval	Ramp Angle Interval	Experimented Configuration Number	Successful Walking Number	Successful Walking Percentage
Training	[3 10]	[− 7° 7°]	1240	398	32.1%
Validation	[2 11]	[− 8° 8°]	900	195	21.67%
Testing	[1 12]	[− 9° 9°]	930	148	15.91%

3.1.1 Torque Control Data Sets

In the first scenario, NNs are assigned to generate torque patterns according to speed excitation input and feedback taken from the biped robot platform, as listed in detail below:

- 6 Output data sequence [Dimensions: 6x1000]
  - Joint torques: T_{ri} where $i = 1, ..., 6$
- 32 Input data sequence [Dimensions: 32x1000]
  - 12 Feedback connections: $Feed_i(\boldsymbol{x}, \dot{\boldsymbol{x}})$ where $i = 1, ..., 12$
  - 4 Limb angle states: θ_i where $i = 1, ..., 4$
  - 14 Velocity states: $\dot{x}_i$ where $i = 1, ..., 14$
  - 1 Speed excitation level
  - 1 Bias term

3.1.2 Position Control Data Sets

In the second scenario, NNs are assigned to generate limb angle patterns according to speed excitation value and ramp angle inputs, as given in detail below:

- 4 Output data sequence [Dimensions: 4x1000]
  - Limb angles: θ_i where $i = 1, ..., 4$
- 3 Input data sequence [Dimensions: 3x1000]
  - 1 Speed excitation value
  - 1 Ramp angle in degree
  - 1 Bias term

Note that, different from the torque control scenario, NN takes ramp angle information as an input because the neural controller needs to run without feedback connections from the robot platform so it does not have the opportunity to observe the walking environment, indirectly.

3.1.3 PID Controller Data Sets

In the third scenario, NNs are assigned to generate torque patterns according to reference limb angles and limb angle

feedback taken from the biped robot, as shown in detail below:

- 6 Output data sequence [Dimensions: 6x1000]
  - Joint torques: T_{ri} where $i = 1, ..., 6$
- 8 Input data sequence [Dimensions: 8x1000]
  - 4 Reference limb angle states: θ_i^{nn} where $i = 1, ..., 4$
  - 4 Limb angle states: θ_i where $i = 1, ..., 4$

Different from two previous data sets, joint torques are outputs of PID controller which is tuned to track CPG driven robot limb angles in a closed loop system. Details of PID controller is given in the related subsection.

3.2 Torque Controller Implementation

In the first control scenario, closed loop NNBCs are assigned to generate joint torque values. Note that in this scenario, the neural controller receives (external) speed excitation level and (internal feedback) biped robot states signals as its inputs, as shown in Fig. 6. In this scenario, NNs are trained via torque control data set which is extracted from CPG driven biped robot model walking data as explained, previously. We intuitively expect that taking feedback directly from biped robot and indirectly from environment should increase the stability of locomotion and robustness against the external disturbances. Thus, larger action space may be obtained while stability is preserved due to the generalization ability of NNBCs.

3.2.1 Neural Network Architecture Design

NN architecture has key importance in control performance. For this reason, different network architectures are trained and tested to find the most appropriate architecture for torque control problem. Since the torque control requires continuous output in an interval, regression layer is added to the end of the NN architecture, as shown in Fig. 5. Moreover, the robot model has a hybrid dynamic structure due to changes between flight and stance phases for each leg. When a foot is not in contact with the ground in the flight phase, ankle joint torque does not affect system dynamics. However ankle joint torque, ground reaction and friction forces affect system dynamics while the foot is in contact with the ground in the stance phase. Therefore, these walking phase changes are needed to be tracked for high-performance control. For this purpose, at least one recurrent layer is decided to be added to NN architecture. By considering these points, five different NN architectures in which different numbers of feedforward and recurrent layer with various neuron numbers are used to find the most appropriate NNBC architecture as reported in Table 3. In addition to these, feedforward layer between LSTM layers is also used to diversify the tested architectures. Finally, all networks are ended with six neurons including feedforward regression layer.

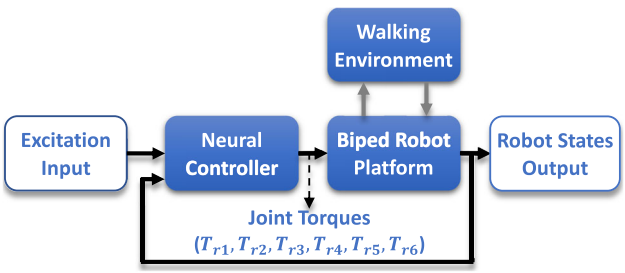


Fig. 6 Torque control diagram

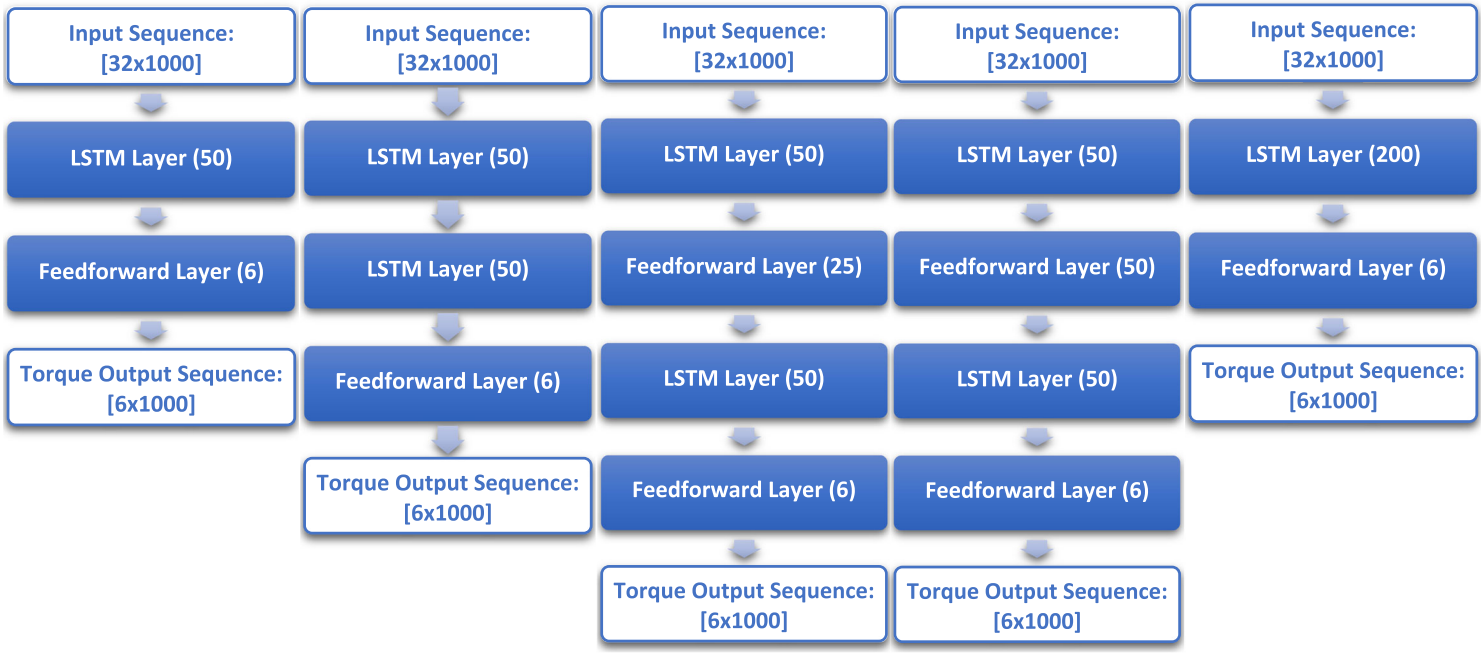
After training, two different tests are applied to measure the competence of these hybrid networks.

In the first test, trained HNN structures are tested to determine their stable locomotion controlling capability over training, validation and testing set configurations in the simulation environment. In detail, neural controllers are tested for all configurations which are used in the forming of data sets independent of the CPG success. In these simulations, bias term, excitation input values and feedback connections taken from the robot model are given to NN as input. Then, output of the neural controller which consists of six torque values is given to the robot model. To avoid overfitting, neural network weights that perform the highest validation set walking success percentages throughout the training process are chosen and they are utilized to find walking success percentages in other data sets. After the simulations, robot model movements are evaluated with successful walking criteria which are determined in data set generation. Then, results are reported in Table 3. Hence our aim is to observe whether the neural controller generates stable locomotion for the cases where CPG is not successful. In the second test, input sequences in torque control training, validation, and test data sets are given to the network, and outputs are compared with the desired output sequence. The mean squared error metric is applied to the difference and reported as approximation error in Table 3. Different from the first test, the second test is applied only for patterns where CPG is successful to generate stable walking while the first test is applied for each configuration of data sets regardless of CPG walking success.

The smallest NN architecture consists of one recurrent layer with 50 neurons and one regression layer with 6 neurons in Configuration #1. Configuration #2 includes two recurrent layers different from Configuration #1. While high number of recurrent layer utilization is beneficial for decreasing training set approximation error, it also decreases the test set walking success percentage. Feedforward layer addition between recurrent layers is tested with

Table 3 NNBCs architecture for torque control scenario. Neuron number of layers are given in parentheses

	Config. #1	Config. #2	Config. #3	Config. #4	Config. #5
Learning constant	0.001	0.001	0.001	0.001	0.001
L_2 constant	0	0	0	0	0
Mini-batch size	1	20	20	20	20
LSTM layer number	1	2	2	2	1
Feedforward layer number	1	1	2	2	1
Feedforward layer activation	Linear	Linear	Linear	Linear	Linear
Regression neuron number	6	6	6	6	6
Neural network structure					



Epoch number	20000	20000	18000	18000	20000
Training set walking success	26.85%	30.16%	24.27%	21.05%	26.21%
Validation set walking success	27.56%	29.78%	19.11%	13.44%	17.44%
Testing set walking success	14.73%	11.08%	6.88%	7.74%	12.9%
Training set approximation error	$5.89 \cdot 10^{-4}$ Nm	$5.06 \cdot 10^{-4}$ Nm	$5.43 \cdot 10^{-4}$ Nm	$5.1 \cdot 10^{-4}$ Nm	$5.15 \cdot 10^{-4}$ Nm
Validation set approximation error	$1.02 \cdot 10^{-3}$ Nm	$1.05 \cdot 10^{-3}$ Nm	$1.13 \cdot 10^{-3}$ Nm	$1.2 \cdot 10^{-3}$ Nm	$8.89 \cdot 10^{-4}$ Nm
Testing set approximation error	$1.02 \cdot 10^{-3}$ Nm	$8.61 \cdot 10^{-4}$ Nm	$8.75 \cdot 10^{-4}$ Nm	$9.17 \cdot 10^{-4}$ Nm	$8.84 \cdot 10^{-4}$ Nm

Configuration #3 and #4. Small number of intermediate feedforward neuron utilization generally become more successful than high number of neuron utilization. Finally, a high number of neuron utilization in single recurrent layer is examined in Configuration #5. It is interesting to see that approximation error and walking success percentage metrics do not show high correlation in data sets. Although Configuration #1 has the highest testing set approximation error, it has the highest testing set walking success rate. Moreover, Configuration #1 can be trained faster than other controller networks because of its small network size. In this context, Configuration #1 is selected as the most appropriate neural controller architecture due to its high testing set walking success rate and small network size. Therefore, this controller architecture is utilized in the continuation of this study.

3.3 Position Controller Implementation

In the second control scenario, NNBCs are assigned to produce limb angles while they are taking speed excitation, ramp angle external inputs. Produced limb angles are tracked by a second closed-loop controller, as shown in Fig. 7. In this scenario, NNs are trained via position control data sets which is extracted from CPG driven biped robot walking data as explained in previous related subsection. Since the NN architecture Configuration #1 in Table 3 is successfully employed in torque control scenario which require relating 32 inputs with 6 outputs, it is reasonable to assume that the same configuration should also perform sufficiently well for position control scenario which include only 3 inputs and 4 outputs.

3.3.1 PID Controller Design

In the position control simulations, reference limb angles (θ_i, $i = 1, 2, 3, 4$) for the limbs of biped robot in Fig. 1 are produced by the neural controller in feedforward path and joint torques (T_{ri}, $i = 1, 2, ..., 6$) are calculated with PID controller as shown in Fig. 7. To achieve this, four limb angle input is associated with six torque output by six PID controllers, which use the same controller parameters, as given in Eqs. 16–24:

$$e_i[t] = \theta_i^{nn}[t] - \theta_i[t], \ for\ i = 1, 2, 3, 4, \tag{16}$$

$$\dot{e}_i[t] = \dot{\theta}_i^{nn}[t] - \dot{\theta}_i[t], \ for\ i = 1, 2, 3, 4, \tag{17}$$

$$se_i[t] = \sum_{i=0}^{t} t(\theta_1^{nn}[i] - \theta_1[i]), \ for\ i = 1, 2, 3, 4 \tag{18}$$

$$T_{r1}[t+1] = I_1(K_p e_1[t] + K_i se_1[t] + K_d \dot{e}_1[t]), \tag{19}$$

$$T_{r2}[t+1] = I_1(K_p e_2[t] + K_i se_2[t] + K_d \dot{e}_2[t]), \tag{20}$$

$$\begin{aligned} T_{r3}[t+1] = {} & I_1(K_p e_1[t] + K_i se_1[t] + K_d \dot{e}_1[t]) \\ & - I_2(K_p e_3[t] + K_i se_3[t] + K_d \dot{e}_3[t]), \end{aligned} \tag{21}$$

$$\begin{aligned} T_{r4}[t+1] = {} & I_1(K_p e_2[t] + K_i se_2[t] + K_d \dot{e}_2[t]) \\ & - I_2(K_p e_4[t] + K_i se_4[t] + K_d \dot{e}_4[t]), \end{aligned} \tag{22}$$

$$T_{r5}[t+1] = \ I_2(K_p e_3[t] + K_i se_3[t] + K_d \dot{e}_3[t]), \tag{23}$$

$$T_{r6}[t+1] = I_2(K_p e_4[t] + K_i se_4[t] + K_d \dot{e}_4[t]), \tag{24}$$

where θ_i^{nn}, $i = 1, 2, 3, 4$ is the output of neural position controller and θ_i, $i = 1, 2, 3, 4$ is the actual limb angles of right upper leg, left upper leg, right lower leg, left lower leg, respectively. I_1 and I_2 are inertia of lower and upper legs, respectively.

First, it is required to determine a suitable performance metrics for tuning controller parameters used in Eqs. 19–24. In our problem, one intuitive performance metric may be considered as approaching to the torque output of CPG ($\boldsymbol{T_r^{CPG}}$) with the PID controllers for each of $n = 398$ training data set patterns as:

$$\begin{aligned} & \min_{K_p, K_i, K_d} \sum_{j=1}^{n} \sum_{t=1}^{N} \frac{\Delta t}{2} (T_1[t] - T_2[t])^2 \\ & \quad \text{s.t.} K_p, K_i, K_d \geq 0 \\ & \text{where } T_1[t] = \boldsymbol{T_r^{PID(j,K_p,K_i,K_d)}}[t] \\ & \qquad T_2[t] = \boldsymbol{T_r^{CPG(j)}}[t] \\ & \qquad \boldsymbol{T_r^{PID}}\text{:Joint torques of PID controller} \\ & \qquad \boldsymbol{T_r^{CPG}}\text{:Joint torques of CPG controller.} \end{aligned} \tag{25}$$

Unfortunately, due to the highly nonlinear structure of biped robot dynamics and CPG, the optimization problem given by (25) is not tractable and not efficient as well for such a tuning process. For this reason, analysis of all walking duration of PID controlled driven biped robot is chosen to determine appropriate performance metrics. To achieve this, desired robot limb angles (θ_i, $i = 1, 2, 3, 4$), which exist in position control training data set as output data sequence, are given as reference input to the PID including closed-loop system instead of neural controller outputs (θ_i^{nn}, $i = 1, 2, 3, 4$) in Fig. 7. After the simulation, walking data is classified as successful or not by using criteria such as minimum displacement, maximum jumping height, falling, etc. By using these ideas, we developed a heuristic metric $f_{successs}$ which tries to maximize the successful walking over the training set patterns, and the associated optimization process could be given as:

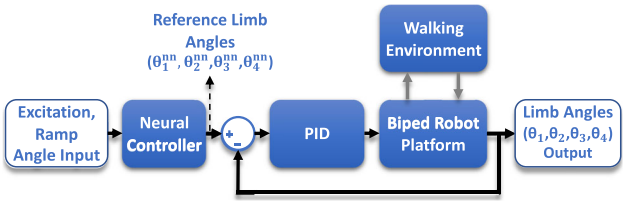


Fig. 7 Position control diagram

$$\max_{K_p,K_i,K_d} \sum_{j=1}^{n} f_{success}(r, \theta^{(j)}, K_p, K_i, K_d)$$
$$\text{s.t.} K_p, K_i, K_d \geq 0 \quad (26)$$
$$\text{where } r: \text{ramp angle}$$
$$\theta^{(j)}: [\theta_1^{(j)}, \theta_2^{(j)}, \theta_3^{(j)}, \theta_4^{(j)}].$$

In this way, the highest success rate is acquired with $K_p = 2000$, $K_i = 400$ and $K_d = 150$ and these parameters are employed in the PID controllers in the remaining part of the study.

3.4 Neural Network Replacement of PID Controller

In the third control scenario, PID controller is replaced with a neural controller as shown in Fig. 8 as an extension to neural position controller experiments in feedforward path. For this purpose, NN architecture Configuration #1 in Table 3 is utilized because of its success in closed-loop torque control problem. The proposed closed-loop neural controller takes four reference limb angle values ($\theta^{nn}[t]$) from the neural controller in the feedforward path and four biped robot platform limb angles ($\theta^{nn}[t-1]$) at time t as input. Then, the neural controller calculates six torque outputs ($T_r[t]$) at time t.

In the training set generation of the closed-loop NN, PID controller is utilized over position control training data set patterns. To this end, reference limb angles, instant biped robot limb angles, and PID torque outputs are recorded for training patterns. After that, four different neural network configurations are trained as shown in Table 4. Among these, LSTM layer or feedforward layers with hyperbolic tangent activation functions are utilized at hidden layers. When feedforward neurons are utilized in hidden layers, neuron numbers are chosen to give similar number of tunable neural network weight to LSTM layer utilization. In the training, Adam optimizer is utilized with a technique similar to early stopping method. Under these conditions, the lowest error rate was obtained with a combination of $L_2 = 0$ and 50 LSTM neuron number. This NN is utilized in position control experiments and results are reported in the later subsection.

It is interesting to see that, different from the torque control problem nonzero L_2 regularization constant resulted in higher training set error and in lower walking control success compared to zero L_2 training configurations. This situation may be related to the inability to learn training set patterns with L_2 regularization.

4 Results and Discussion

In this section, proposed NNBCs' stable locomotion controlling capacity with biped robot platform is evaluated under varying ground conditions and various metrics. Then, NNBCs are compared with each other and CPG controller which is utilized in training set generation. Finally, the simulation results are discussed.

4.1 Torque Control Simulations

Generalization capability limits of proposed NNBC architecture are investigated with the utilization of different L_2 regularization constants and mini-batch sizes at the training stage. Mini-batch size (M) means the number of patterns that are utilized to update network weights together. L_2 regularization is a weight decay method that is employed to improve the generalization capability of the network, see e.g. [70]. Employed mini-batch sizes and L_2 regularization constants are given in Table 5. NN parameters are recorded at specific epoch intervals during training. After the completion of the training stage, the recorded network parameters are used to measure the capability of stable gait generation on the training, validation and testing data set configurations.

Reported success rates in Table 5 are obtained by performing walking simulation for each ramp angle and speed excitation value combinations given in Table 2. For this reason, reported success rates for each data sets needs to be compared by CPG values in Table 2 while concluding about the superiority of proposed NNBC architectures. To this end, the NN training configurations that achieved higher success than the CPG are marked in bold in Table 5. Note that CPG is utilized in obtaining the data set with which the NNs are trained. Clearly, the higher performance of neural controllers as compared to CPG is a result of their generalization ability.

The highest validation set walking success percentage is obtained for $L_2 = 0.5$ and $M = 199$ configuration. The NN controlled walking success rate changes throughout the training process as shown in Fig. 9. One reason for it could be the possible over-fitting problem in the later stages of training. To avoid this problem, the NN weights which provide the highest validation set walking success rate are chosen from recorded network weights for each training

configuration. Then, selected network weights are employed to reach reported walking success percentages on training, validation and testing data sets in Table 5. Thus, the problem of over-fitting is avoided by applying a technique similar to early stopping.

The training configuration of $L_2 = 0.5$ and $M = 199$ reached the highest success in all data sets at the 4000th epoch. Therefore, the NN which uses these weights is found as the best torque controller and it is referred to as "selected controller" in the remaining part of the subsection. To examine the performance of the selected controller, mean squared error between torque outputs of CPG and the selected controller is given for each training, validation, and testing data set patterns in Fig. 10. When Fig. 10 is examined, it is observed that the difference between the two controller outputs is higher for patterns at the outer boundaries of the data sets. It is also observed that the selected controller is more sensitive to the change of ramp angle than the speed excitation value changes. The difference between the two controllers has a tendency to increase for combinations of high speed excitation and ramp angle values.

In order to understand the effect of observed differences between controllers on the locomotion controlling performance, walking tests are performed in the simulation environment for ramp angle and speed excitation configurations shown in Fig. 11. As can be seen in Fig. 11, the selected controller has higher success than CPG in all data sets. In detail, the selected controller reaches higher walking success compared to the CPG for downhill ramp angles in the training, validation and test data sets. Similarly, the selected controller shows higher success than CPG for uphill ramp angles in the training and testing data sets. CPG shows higher success for uphill ramp angles than the selected controller in the validation data set. In this context, increasing difference between selected controller and CPG for patterns at the outer boundaries of data sets in Fig. 10 apparently contributes positively to the walking control performance.

To evaluate the noise rejection performance of the selected controller, walking success of the selected controller and CPG have been compared under rough terrain conditions. For this purpose, zero mean and unit variance Gaussian distribution is employed in the generation of the random numbers to make the rough terrains used in the simulation environment. Nine different walking surfaces with varying roughness levels have been modeled by using the same random numbers in order to ensure equal conditions between experiments. To achieve this, random numbers are scaled with the multipliers in [0 0.001 0.002 0.005 0.01 0.015 0.02 0.03 0.04] array. Then, scaled random numbers are added to the height of the walking surface with 0.1 meter wide intervals to obtain rough terrain. After that, produced rough terrain models are added to the inclined ground model on which selected controller and CPG driven robot moves. The variation of the walking success of the robot platform which is driven by the selected neural controller and CPG operated under these conditions depending on the level of ground roughness is shown in Fig. 12 and Table 6. For each roughness level, the selected controller reached equal or higher walking success rates than CPG at all data sets and higher

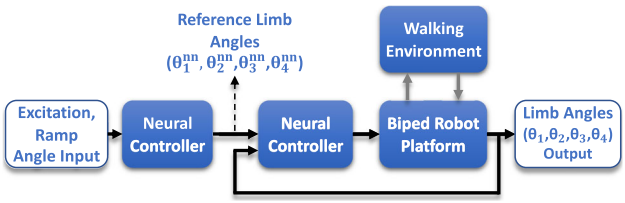


Fig. 8 Position control diagram

Table 4 Neural replacement of PID controller trials

	Config. #1	Config. #2	Config. #3	Config. #4
Learning constant	0.001	0.001	0.001	0.001
L_2 constant	0	0.05	0	0
Mini-batch size	199	199	199	199
Hidden layer number	1	1	2	4
Neuron number at each hidden layer	50	50	103	61
Hidden layer neuron type	LSTM	LSTM	FNN (tanh)	FNN (tanh)
Regression neuron number	6	6	6	6
Epoch number	51000	60000	15000	10000
Training set walking success percentage	75.4%	0%	39.27%	31.85%
Validation set walking success percentage	68%	0%	35.44%	33.22%
Testing set walking success percentage	55.48%	0%	30.22%	22.37%
Training set approximation error (Nm)	$5.17 \cdot 10^{-5}$	$3.7 \cdot 10^{-4}$	$5.2 \cdot 10^{-4}$	$5.2 \cdot 10^{-4}$
Validation set approximation error (Nm)	$5.47 \cdot 10^{-5}$	$3.76 \cdot 10^{-4}$	$5.26 \cdot 10^{-4}$	$5.26 \cdot 10^{-4}$
Testing set approximation error (Nm)	$5.21 \cdot 10^{-5}$	$3.59 \cdot 10^{-4}$	$5.1 \cdot 10^{-4}$	$5.11 \cdot 10^{-4}$

Table 5 NN driven walking success comparison for torque control scenario

	Data set	Mini-batch size		
		M = 1	M = 40	M = 199
$L_2 = 0$	Training	29.03%	27.9%	30.4%
	Validation	**24.44%**	14.11%	18.44%
	Testing	11.51%	10.32%	9.68%
$L_2 = 0.0005$	Training	28.06%	–	–
	Validation	19.44%	–	–
	Testing	12.04%	–	–
$L_2 = 0.005$	Training	28.55%	30.73%	24.44%
	Validation	**28.11%**	18.56%	17.67%
	Testing	**17.96%**	15.16%	11.29%
$L_2 = 0.05$	Training	14.76%	**35.65%**	**48.47%**
	Validation	**35%**	13.44%	**22.33%**
	Testing	**24.41%**	15.81%	**23.66%**
$L_2 = 0.5$	Training	4.19%	**44.35%**	**53.06%**
	Validation	10.89%	**30.44%**	**31.22%**
	Testing	2.69%	**25.91%**	**33.55%**

success rates are marked in bold in Table 6. As shown in Fig. 12, both controllers form a monotonically decreasing walking success for roughness multipliers of 0.002 and above. It is seen that the selected neural controller and CPG cannot perform successful locomotion on any data set for roughness multipliers of 0.04 and above.

4.2 Position Control Simulations

Generalization capability limits of NN controller in the feed-forward path architecture are investigated with the utilization of different L_2 regularization constants and mini-batch sizes in the training process. Employed mini-batch sizes and L_2 regularization constants are given in Table 7. NN parameters are recorded at specific epoch intervals during training. After the completion of the training stage, the recorded network parameters are used to measure the successes of stable gait controlling in training, validation and testing data sets. During locomotion tests, PID controller is utilized in the closed loop.

The highest validation set walking success rate is obtained for $L_2 = 0.05$ and $M = 199$ configuration. The NN controlled walking success rate changes through the training process as shown in Fig. 13. As compared to Fig. 9, now we have more oscillatory behaviour in the success rate change during training. To avoid over-fitting possibility the network weights which provide the highest validation set walking success rate are chosen from recorded network weights for each training configuration. After that, selected network weights are used to obtain walking success percentages on training, validation and testing data sets given in Table 7. In Table 7 the NN training configurations that achieved higher success than the CPG are marked in bold. Note that CPG is utilized in obtaining the training data set. Clearly, the better performance of neural controllers as compared to CPG is a result

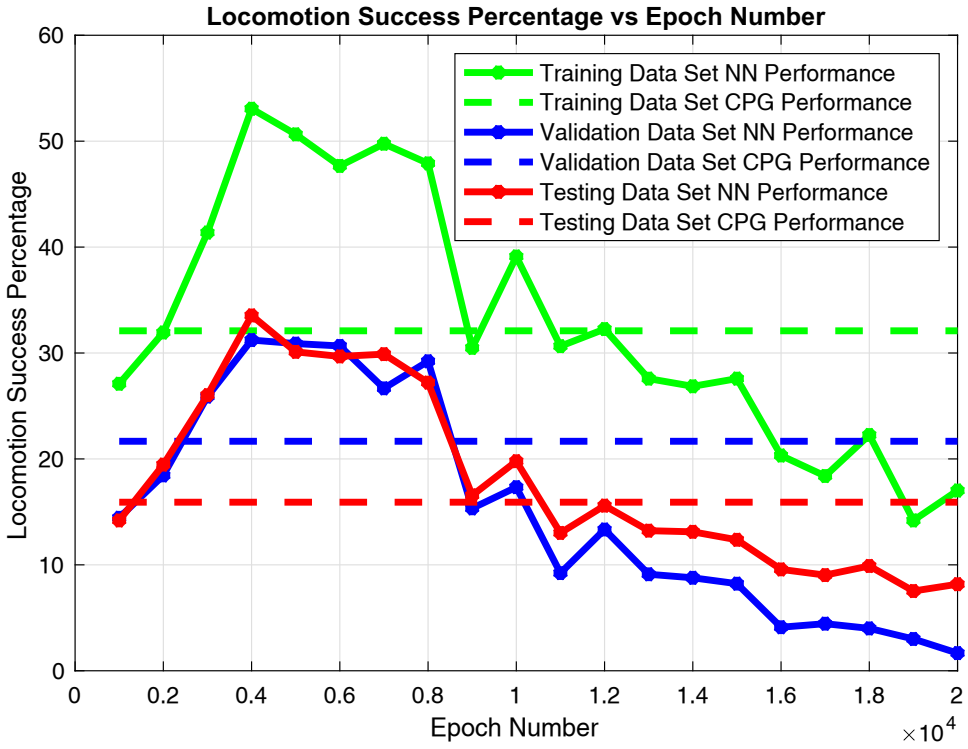
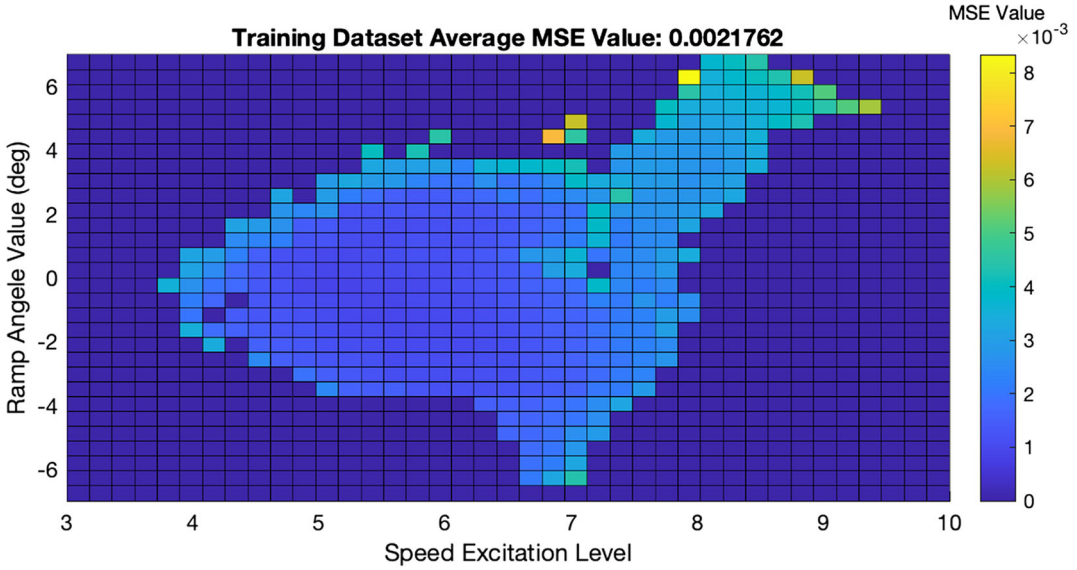
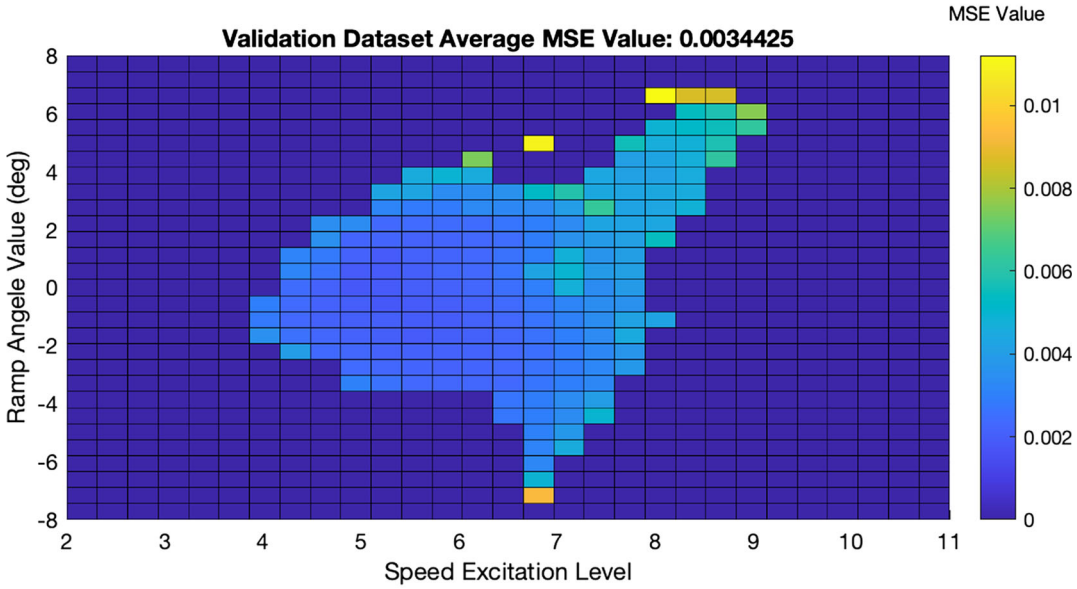
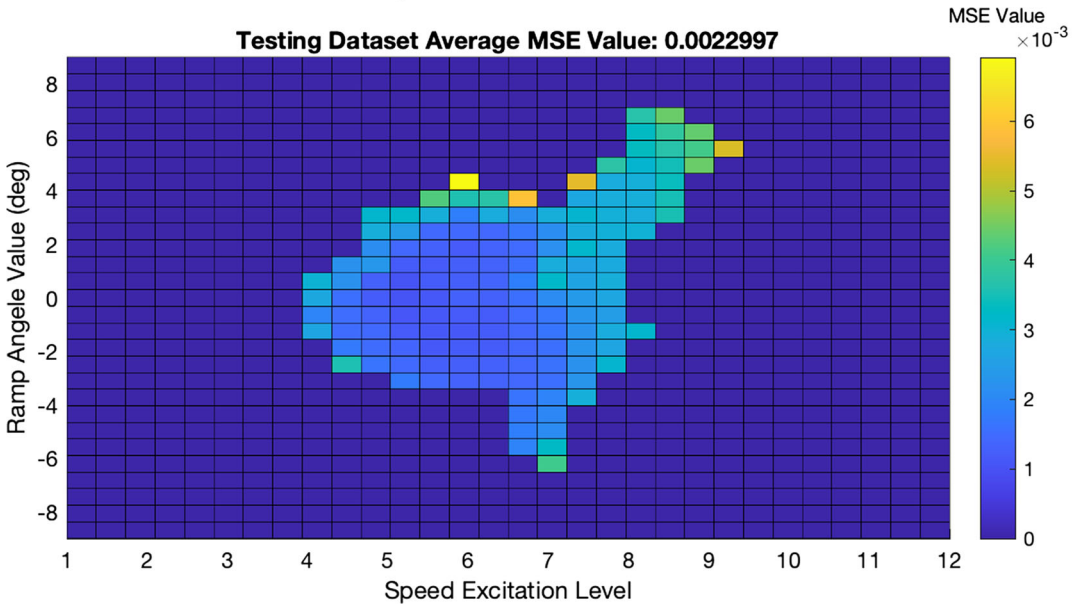


Fig. 9 Walking success change of selected neural controller on data sets during training


Training Dataset Average MSE Value: 0.0021762
MSE Value
×10^-3
Ramp Angele Value (deg)
Speed Excitation Level


Validation Dataset Average MSE Value: 0.0034425
MSE Value
Ramp Angele Value (deg)
Speed Excitation Level


Testing Dataset Average MSE Value: 0.0022997
MSE Value
×10^-3
Ramp Angele Value (deg)
Speed Excitation Level

◀ **Fig. 10** Calculated torque output difference between CPG and the selected controller for all patterns in each data set

of their generalization ability over all data sets. Therefore, NN which uses weights for configuration of $L_2 = 0.05$ and $M = 199$ at the 37500th epoch is the best position controller and it is referred to as "selected controller" in the remaining part of the subsection.

The selected controller has higher success than CPG in all data sets as shown in Fig. 14. In detail, the selected controller reaches higher walking success compared to the CPG for uphill and downhill ramp angles in all data sets. Similarly, the selected controller also shows higher success than CPG for high and low speed excitation values in all data sets. It is observed that for some speed excitation values in the range of [6 7.5] selected controller failed as can be seen in Fig. 14. When the reasons for this behavior are examined, it is seen that for these speed excitation values robot platform shows similar behaviors to jumping instead of walking. This behavior may possibly be due to PID controller structure used to control robot platform. This problem may possibly be solved by optimizing PID controller given in equations between Eqs. 16–24.

4.3 PID Replacement Simulations

In the third control scenario, PID controller is replaced with a NNBC as an extension to the second scenario. Thus, selected controller in feedforward path is utilized together with closed loop NNBC instead of PID controller.

In order to analyze the robustness of proposed NNBC, CPG controller, selected controller with PID (NN+PID; Fig. 7) and selected controller with closed loop neural controller (NN + NN; Fig. 8) have been tested on rough terrain conditions which are the same with the torque control rough terrain experiments and results are given in Table 8. For each roughness level, the selected controller with configuration in Figs. 7 and 8 reached equal or higher walking success rates than CPG at all data sets. Among two configurations of the selected controller, closed loop neural controller utilization in Fig. 8 mostly reaches higher success rates for roughness multiplayer of 0.01 and below. After this level, PID utilization as closed loop controller in Fig. 7 generally give better results on data sets. Here we note that NN is trained with flat terrain data, but nevertheless it still shows acceptable performance for rough terrain walking. This behavior may also be related to the generalization ability of NNBCs.

When controller performance on position control scenario in Figs. 7 and 8 and closed loop controller performance on torque control scenario in Fig. 6 are compared with each other, position control structure given in Fig. 8 performs better than the other configurations. In addition to this, for the high roughness multiplier values position control structure given in Fig. 8 is less successful compared to PID controller.

4.4 Controller Sensitivity Analysis

In most of the control applications, there are differences between the mathematical model which is used to design the controller and the actual plant that needs to be controlled so good controllers should work in a harmonious and resistant way against small variation in plant dynamics and external disturbance, respectively. In this subsection, sensitivity analysis of proposed NNBCs is reported against changing robot weight in the simulation environment. To this end, it is decided to increase the weight of each mass on the robot model and compare the resulting walking success rates for CPG and proposed NNBCs. This analysis can be thought of as another way of checking the existence of over-fitting to robot model dynamics in the simulation environment.

During this analysis, NNBCs are not retrained with weighted robot models to satisfy fair comparison. In order to analyze the sensitivity of proposed NNBCs, CPG controller, selected controller for torque control scenario (NN; Fig. 6), selected controller with PID for position control scenario (NN + PID; Fig. 7) and selected controller with closed loop neural controller for position control scenario (NN + NN; Fig. 8) have been tested with increasing robot model weight and obtained results are given in Table 9. For each data set and robot weight increase level, the highest walking success rates are marked in bold in Table 9.

For each weight level, the selected NNBCs reached higher walking success rates than CPG at all data sets. Among three configurations of the selected controllers, the selected controller with closed-loop neural controller for position control scenario (NN + NN; Fig. 8) reaches the highest training and validation data set performances. Later, selected controller with PID for position control scenario (NN + PID; Fig. 7) reaches the highest testing data set performance. Finally, the selected controller for the torque control scenario (NN; Fig. 6) outperforms CPG for each weight increase percentage but it reaches lower walking success rates than position controllers similar to rough terrain experiment results. These results may be related to the generalization ability of NNBCs.

4.5 Joint Torque Limitation Analysis

Successful real-life implementations of legged locomotion controller algorithms have to be able to work in a harmonious way with various physical constraints of robot plant such as actuator torque and speed limitations. For this reason, it is important for a locomotion control algorithm to be able to work under some constraints with adequate performance. To this end, we applied torque optimization to proposed NNBCs

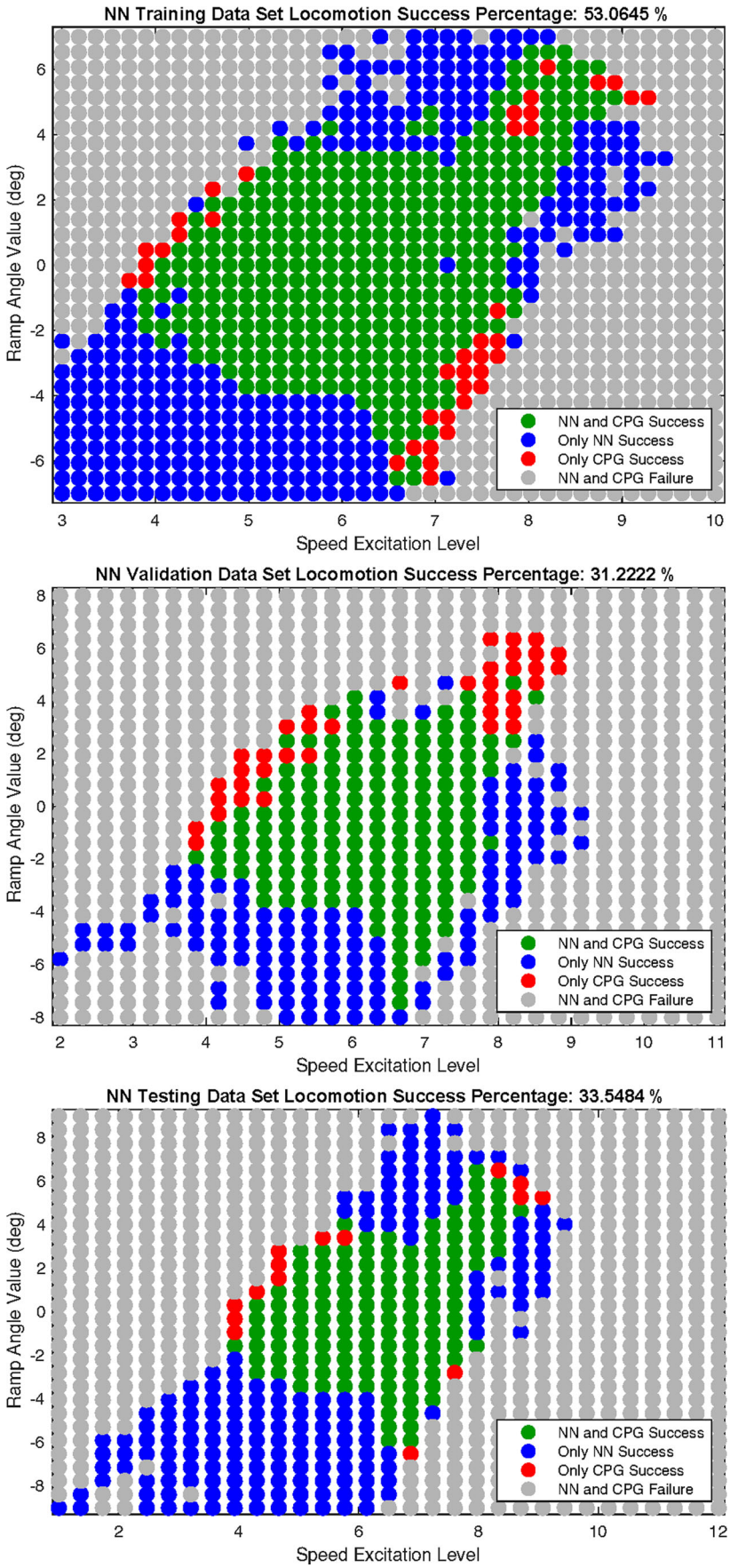


Fig. 11 Successful locomotion generation capability of selected controller and CPG for different ramp angle and speed excitation value combinations in the data sets

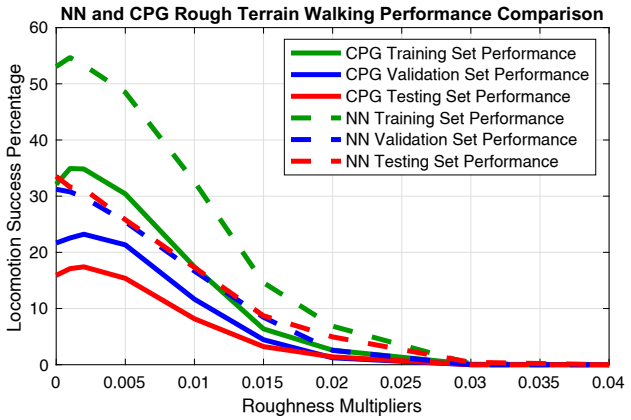


Fig. 12 Successful walking percentages of the selected controller and CPG for different roughness multipliers

and we tested CPG and NNBCs under torque limitation which is the 25% of the highest joint torques generated by the CPG controller during generation of the training data set.

Table 6 Rough terrain torque control experiments

Roughness	Data	Configuration	
Multiplier	Set	CPG	NN
0	Training	32.1%	**53.06%**
	Validation	21.67%	**31.22%**
	Testing	15.91%	**33.55%**
0.001	Training	34.92%	**54.68%**
	Validation	22.56%	**30.78%**
	Testing	17.1%	**31.61%**
0.002	Training	34.84%	**53.31%**
	Validation	23.22%	**29.78%**
	Testing	17.42%	**31.29%**
0.005	Training	30.4%	**48.47%**
	Validation	21.33%	**25.44%**
	Testing	15.38%	**25.81%**
0.01	Training	17.42%	**32.58%**
	Validation	11.67%	**16.67%**
	Testing	8.17%	**17.31%**
0.015	Training	6.37%	**14.6%**
	Validation	4.44%	**8.44%**
	Testing	3.23%	**8.71%**
0.02	Training	2.58%	**6.85%**
	Validation	1.22%	**2.56%**
	Testing	1.4%	**4.95%**
0.03	Training	0.08%	**0.16%**
	Validation	0%	0%
	Testing	0%	**0.43%**
0.04	Training	0%	0%
	Validation	0%	0%
	Testing	0%	0%

Table 7 NN driven walking success comparison for position control scenario

	Data set	Mini-batch size	
		M = 40	M = 199
$L_2 = 0$	Training	**59.76%**	**49.03%**
	Validation	**46%**	**45.56%**
	Testing	**41.61%**	**34.84%**
$L_2 = 0.005$	Training	**62.1%**	**53.15%**
	Validation	**53%**	**44.56%**
	Testing	**37.96%**	**40.75%**
$L_2 = 0.05$	Training	**68.87%**	**71.69%**
	Validation	**56.22%**	**64.67%**
	Testing	**38.39%**	**55.16%**
$L_2 = 0.5$	Training	**39.35%**	**69.84%**
	Validation	**22.67%**	**39.56%**
	Testing	14.62%	**31.08%**

To make a fair comparison, CPG is run for training, validation, and testing set ramp angle and excitation value combinations under the aforementioned torque limitation, and resulting success rates are given in Table 10. For the same purpose, the selected controller for the torque control scenario (NN; Fig. 6) is trained for mini-batch size 199 and various L_2 regularization constants by using the original training set patterns which are truncated with respect to torque limit again. Later on, the selected controller with PID for the position control scenario (NN + PID; Fig. 7) is optimized by reselecting PID coefficients. To this end, optimization in Eq. 26 is repeated under torque limitation. In this way, the highest success rate is acquired with $K_p = 4500$, $K_i = 0$, and $K_d = 600$. After that, the selected controller with closed-loop neural controller for position control scenario (NN + NN; Fig. 8) is trained for mini-batch size 199 by using the original training set patterns which are truncated with respect to torque limit again. Finally, optimized NNBCs were tested under torque limitation and resulting success rates are reported in Table 10. The NNBCs that achieved higher success than the CPG are marked in bold in Table 10.

Under the 25% joint torque limitation, CPG and selected controller for torque control scenario encounter a decline in gait control success but neural controller outperformed the CPG for L_2 regularization constant 0.05 over all data sets. Interestingly, the selected controller with PID for position control scenario reaches even higher success rates than its unlimited joint torque performances. This success increase may be related to the under-optimized situation of the PID controller. Similar to torque control scenario, the selected controller with closed-loop neural controller for

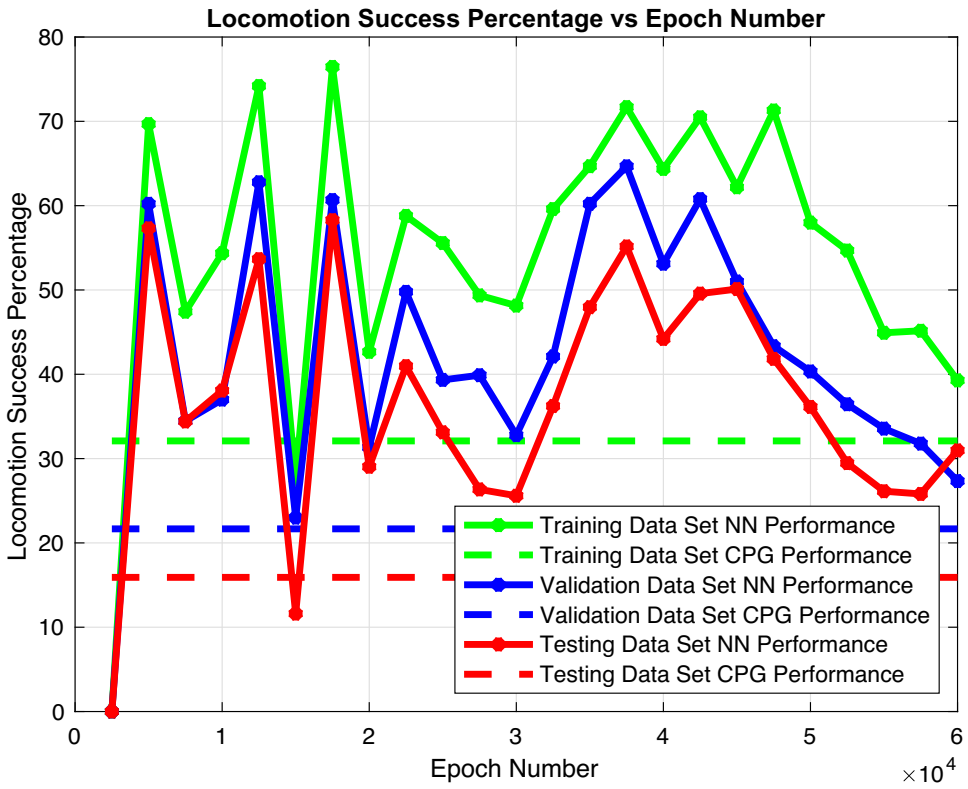


Fig. 13 Walking success change of selected neural controller on data sets during training

position control scenario encounters a decline in walking control success but neural controller outperformed the CPG again. Based on this data, it may be concluded that proposed NNBCs outperform CPG for 25% joint torque limitation constraint and they may work under constraints with various success rates.

4.6 Stability Analysis

Bipedal locomotion stability analysis is a complex problem due to utilized highly nonlinear robot models and diversity of gait types. For this reason, various ways of measuring walking stability are proposed in the literature [8–11, 71]. These methods generally impose some artificial constraints to walking at various levels such as static walking. Unfortunately, these constraints may negatively affect the performance of locomotion in terms of speed, efficiency, disturbance rejection and etc. metrics [71].

To avoid these side effects, we classify stable locomotion patterns by measuring average walking velocity, hip height and detecting falling events as explained in Subsection 3.1. In this way, the least amount of artificial constraints has been imposed on the walking in this study. Even though center of gravity (COG), zero moment point (ZMP), and limit cycle analyses impose some artificial constraints to gait and some of them cannot be fully applicable because of point feet of the employed robot model in this study, they still may help us to understand the behavioral characteristic of proposed NNBCs. For this reason, we utilize COG, ZMP, and limit cycle analyses in this subsection.

The employed robot model has point feet so support polygon does not exist in the single support phase. To make analysis similar to COG and ZMP, we redefine support polygon as the distance between horizontal projections of feet onto the flat surface independently from ground contact. Even under this assumption, support polygon shrinks to a single point when feet are aligned vertically. With this assumption, we track COG and ZMP throughout walking and calculate the percentage of walking duration that satisfies COG and ZMP stability criteria. Table 11 lists COG and ZMP stability criteria satisfied walking duration percentages for CPG and proposed NNBCs on each data set. It is important to note that these analyses are performed only for successful walking patterns.

In terms of Table 11, proposed NNBCs have a lower COG stability criteria satisfying percentage than CPG. Among NNBCs, position controllers satisfy COG stability criteria for a slightly lower percentage than torque controller. Based on this finding, it can be concluded that NNBCs may perform less statically stable locomotion compared to CPG and this situation can be related to the increasing locomotion success of NNBCs. Unexpectedly, ZMP stability criteria are fulfilled for slightly shorter durations than COG as shown

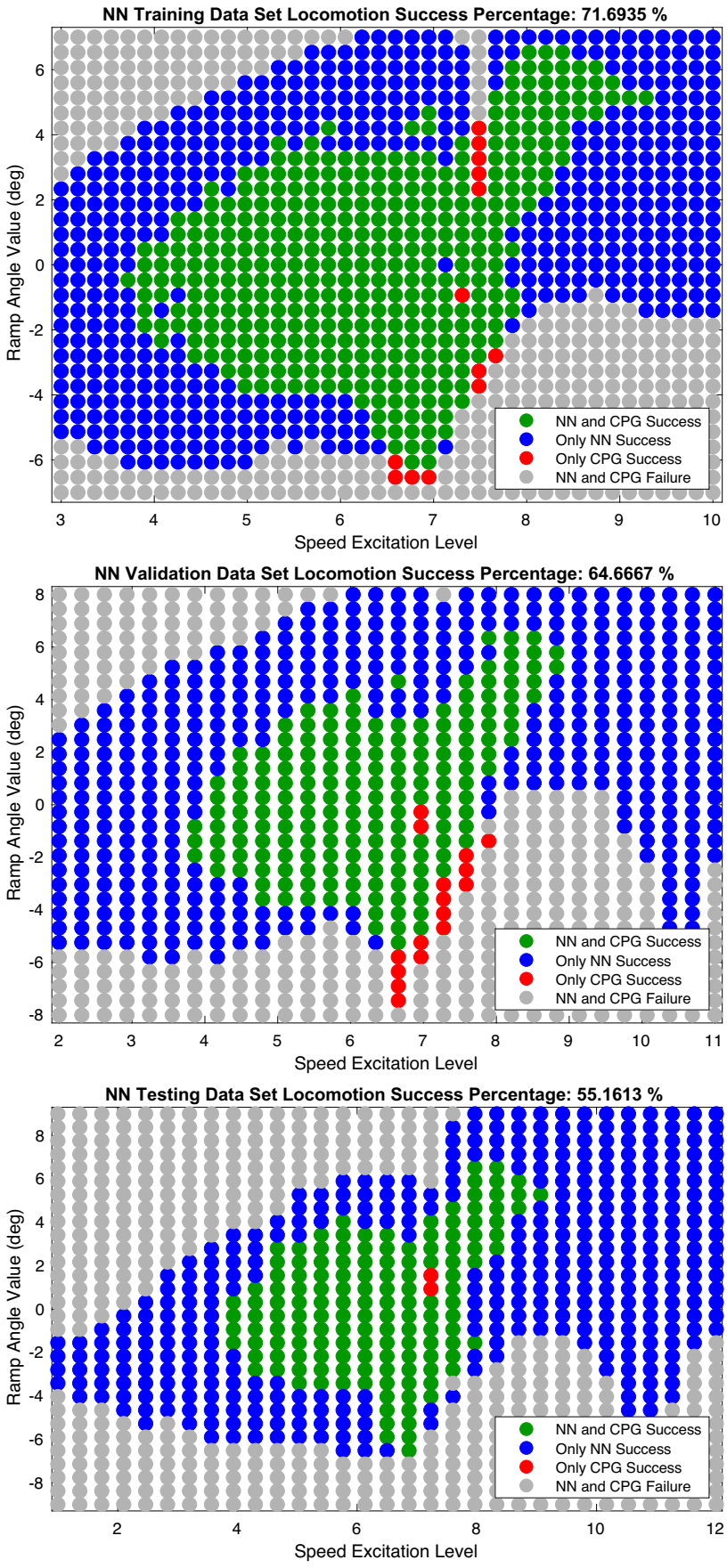


Fig. 14 Successful locomotion generation capability of selected controller and CPG for different ramp angle and speed excitation value combinations in the data sets

Table 8 Rough terrain position control experiments

Roughness Multiplier	Data Set	Configuration		
		CPG	NN + PID	NN + NN
0	Training	32.1%	71.69%	**75.4%**
	Validation	21.67%	64.67%	**68%**
	Testing	15.91%	55.16%	**55.48%**
0.001	Training	34.92%	71.94%	**75.4%**
	Validation	22.56%	64.33%	**67.89%**
	Testing	17.1%	54.84%	**55.38%**
0.002	Training	34.84%	71.53%	**74.76%**
	Validation	23.22%	64%	**67.33%**
	Testing	17.42%	**54.52%**	54.09%
0.005	Training	30.4%	69.44%	**73.06%**
	Validation	21.33%	61.78%	**65.78%**
	Testing	15.38%	**54.19%**	52.26%
0.01	Training	17.42%	62.98%	**67.26%**
	Validation	11.67%	57.78%	**58.89%**
	Testing	8.17%	**49.78%**	47.1%
0.015	Training	6.37%	56.05%	**58.06%**
	Validation	4.44%	**52%**	48.11%
	Testing	3.23%	**41.94%**	40%
0.02	Training	2.58%	**49.6%**	44.35%
	Validation	1.22%	**41.11%**	37.89%
	Testing	1.4%	**35.91%**	31.94%
0.03	Training	0.08%	**25.24%**	23.31%
	Validation	0%	**22%**	19.67%
	Testing	0%	11.61%	**17.2%**
0.04	Training	0%	0%	0%
	Validation	0%	0%	0%
	Testing	0%	0%	0%

Table 9 Robot weight increase experiments

Robot Weight Increase	Data Set	Configuration			
		CPG	Torque Controller (NN)	Position Controller (NN + PID)	Position Controller (NN + NN)
0%	Training	32.1%	53.06%	71.69%	**75.4%**
	Validation	21.67%	31.22%	64.67%	**68%**
	Testing	15.91%	33.55%	55.16%	**55.48%**
2%	Training	31.29%	54.11%	71.05%	**75%**
	Validation	21.78%	29.89%	63.56%	**67.67%**
	Testing	15.81%	32.9%	**54.84%**	54.73%
4%	Training	31.13%	54.11%	70.16%	**74.44%**
	Validation	20.67%	28.22%	62.89%	**67%**
	Testing	14.62%	31.29%	53.01%	**53.44%**
5%	Training	30%	53.15%	69.52%	**74.84%**
	Validation	20.89%	27.89%	62.33%	**66.89%**
	Testing	14.84%	31.4%	52.37%	**52.9%**
6%	Training	29.35%	54.44%	68.79%	**74.27%**
	Validation	19.67%	25.22%	61.33%	**67.11%**
	Testing	14.19%	30.86%	**52.58%**	51.94%
8%	Training	27.74%	52.34%	67.74%	**73.63%**
	Validation	18.89%	24%	61.11%	**65.44%**
	Testing	13.76%	28.71%	**50.75%**	49.78%
10%	Training	26.37%	50.73%	67.02%	**72.42%**
	Validation	18%	22.22%	59.44%	**64.22%**
	Testing	13.23%	26.67%	**49.68%**	48.71%
15%	Training	20.4%	43.47%	63.47%	**69.27%**
	Validation	13.56%	16.89%	55.22%	**61%**
	Testing	9.68%	23.23%	**46.02%**	43.66%
20%	Training	14.52%	33.63%	57.66%	**64.19%**
	Validation	10.56%	13.78%	50.56%	**55.11%**
	Testing	7.31%	17.63%	**41.83%**	37.63%
25%	Training	9.03%	27.5%	47.74%	**53.71%**
	Validation	6.11%	9.11%	42.89%	**47.22%**
	Testing	4.41%	14.09%	**35.38%**	30.43%

in Table 11. When we seek the possible reasons for it we determine that ground reaction forces generated at the stance phase of the associated leg include high amplitude fluctuations and this situation causes noise in the second derivatives of state variables which are utilized in the calculation of ZMP. When trajectories of ZMP are analyzed it is seen that ZMP oscillates to the back and front of the support polygon and does not diverge from the support polygon permanently. The highest ZMP percentages are observed for torque controller NNBCs and the lowest percentages are acquired for position controller NNBCs. Unfortunately, noise arising from ground reaction forces prevents making further analysis for our control scenarios.

Another method of determining the stability of legged locomotion is well-known limit cycle analysis. It imposes fewer constraints to the gait compared to COG and ZMP methods, [71]. Thus, more efficient and natural gaits can be classified as stable walking. In detail, trajectories of the state variables in the successive steps generate closed trajectories which are called limit cycles in state space, see e.g. [13, 71].

To analyze the characterization of limit cycles, the Poincaré section which is a subset of system states at step n is taken as follows for our case:

$$\boldsymbol{h}[n] = [x_2[n], \dot{x}_1[n], \dot{x}_2[n]]^T. \tag{27}$$

Here, we take Poincaré section when the right thigh is in front of the hip and the hip is in the highest position during a step. We describe this combination as an apex point. Poincaré sections of successive apex points can be mapped by a stride function '$\boldsymbol{S}(.)$' as shown below:

Table 10 Joint torque limitation experiments

Data Set	Configuration						
	CPG	Torque Controller (NN)				Position Controller (NN + PID)	Position Controller (NN + NN)
		($L_2 = 0$)	($L_2 = 0.005$)	($L_2 = 0.05$)	($L_2 = 0.5$)	($L_2 = 0.05$)	($L_2 = 0$)
Training	26.77%	**27.5%**	**27.18%**	**37.58%**	6.13%	**92.26%**	**51.94%**
Validation	18.78%	13%	**21.67%**	**22.11%**	3.78%	**83.11%**	**49.22%**
Testing	15.59%	7.74%	**21.29%**	**26.67%**	2.04%	**68.39%**	**40.54%**

Table 11 Center of gravity (COG) and zero moment point (ZMP) analysis for NN driven walking scenarios

	Data set	COG	ZMP
CPG	Training	88.23%	47.14%
	Validation	88.02%	47.40%
	Testing	88.45%	47.47%
Torque	Training	86.77%	50.69%
Controller	Validation	84.38%	52.38%
(NN)	Testing	85.41%	50.29%
Position	Training	80.25%	43.77%
Controller	Validation	83.08%	42.63%
(NN + PID)	Testing	76.50%	39.04%
Position	Training	82.27%	42.00%
Controller	Validation	81.35%	39.99%
(NN + NN)	Testing	76.15%	36.62%

$$\boldsymbol{h}[n+1] = \boldsymbol{S}(\boldsymbol{h}[n]). \tag{28}$$

For a stable periodic motion that converges to a limit cycle, there are fixed points of $\boldsymbol{S}$ function such as $\boldsymbol{h}^*$ and it repeatedly passes from these fixed points between consecutive steps which satisfy the following:

$$\boldsymbol{h}^* = \boldsymbol{S}(\boldsymbol{h}^*). \tag{29}$$

Then stability of the limit cycle can be determined by linearizing function $\boldsymbol{S}$ around the fixed point $\boldsymbol{h}^*$ as given below:

$$\boldsymbol{S}(\boldsymbol{h}^* + \Delta_{\boldsymbol{h}}) = \boldsymbol{h}^* + \boldsymbol{D}\Delta_{\boldsymbol{h}}, \tag{30}$$

where $\Delta_{\boldsymbol{h}} = [\Delta_{x_2}, \Delta_{\dot{x}_1}, \Delta_{\dot{x}_2}]^T$ is a small deviation vector and $\boldsymbol{D}$ is the Jacobian matrix which consists of partial derivatives of $\boldsymbol{S}$ function with respect to states variables in the Poincaré section. If the eigenvalues of the Jacobian matrix are found within the unit circle, it means the limit cycle is locally stable.

Unfortunately, the highly nonlinear model of the biped robot model and controllers do not let us calculate the Jacobian analytically. Under these conditions, a popular way of evaluating the Jacobian is using numerical methods, see e.g. [13, 71, 72]. The partial derivatives in Jacobian matrix $\boldsymbol{D}$ are calculated with numerical methods as explained in [72]. As an example of this approach, we considered a simple successful walking configuration which is 0 degree ramp angle and 6.41 excitation value. In this way, limit cycles of the CPG and NNBCs driven biped robot models for the selected walking patterns, which are shown in the Fig. 15, are analyzed, and eigenvalues of corresponding Jacobian matrices are reported in Table 12. All eigenvalues are found in the unit circle as expected and this is in line with the results of the rough terrain experiments. Moreover, eigenvalues with the lowest amplitude are found in CPG which may be interpreted that CPG shows faster recovery characteristics against applied perturbation. This observation is also in line with the stability expectations stated in [27], but such a limit cycle calculation was not performed there. Among NNBCs, torque controller has the highest eigenvalue amplitude and this is in line with the observation of slow recovery and lower walking success rates than position controller NNBCs.

Table 12 Computed eigenvalues via the limit cycle analysis for success walking patterns obtained by each controller

	Configuration			
	CPG	Torque Controller (NN)	Position Controller (NN + PID)	Position Controller (NN + NN)
Eigen values	−0.0954	0.585	−0.3113	$-0.3790 - 0.2750i$
	0.13	−0.0913	0.0857	$-0.3790 + 0.2750i$
	0.0502	−0.0062	0.0062	0.0008

Note that these eigenvalues are computed for walking 0 degree ramp angle and 6.41 excitation value walking conditions. All controllers acquire successful locomotion for this ramp angle and excitation value pair

4.7 Discussion

The over-fitting issue is undesired and needs to be avoided to increase the real performance of the machine learning algorithms. To avoid this, three separate data sets named

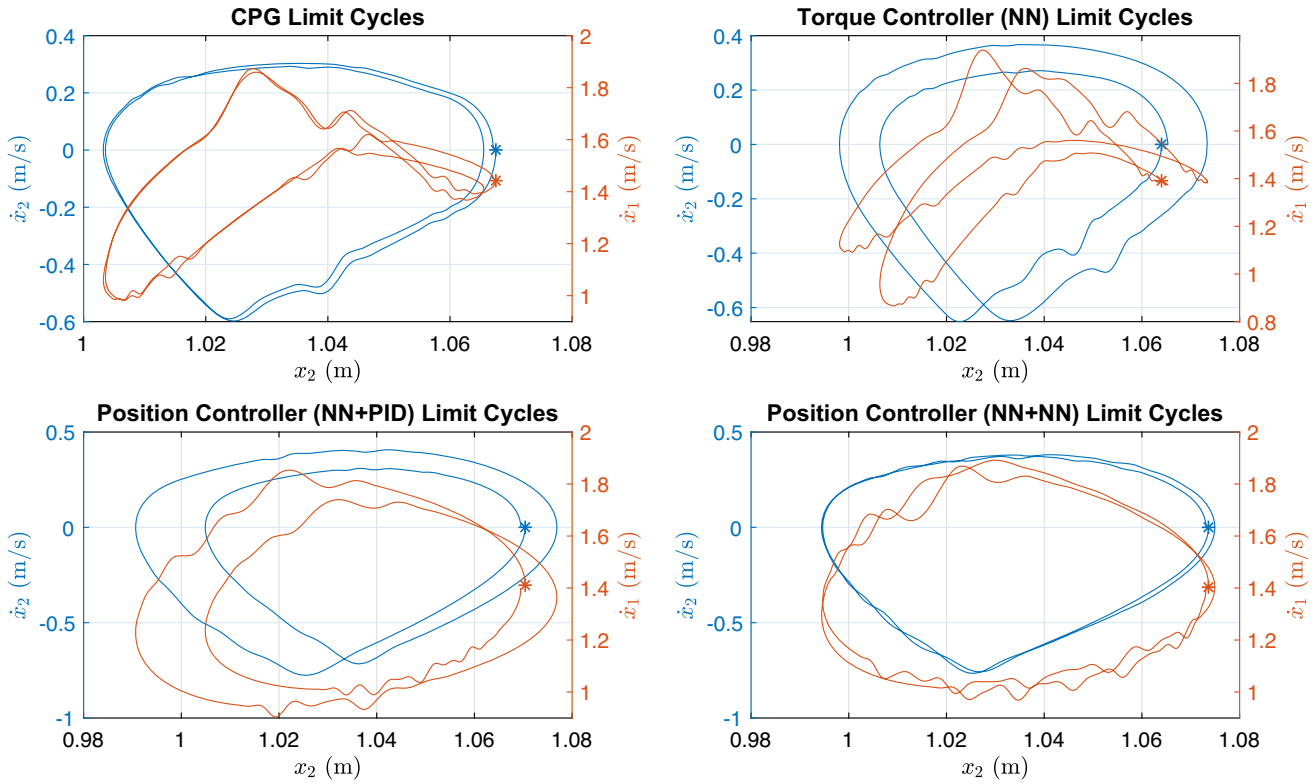


Fig. 15 Limit cycle trajectories of CPG and proposed NNBCs driven biped robot model hip states. Blue and orange lines show (x_2, $\dot{x}_2$) and (x_2, $\dot{x}_1$) limit cycle trajectories, respectively. Note that these limit cycles are generated for 0 degree ramp angle and 6.41 excitation value. All controllers acquire successful locomotion for this ramp angle and excitation value pair

as training, validation and test sets are produced by using CPG driven robot model. While producing these sets, speed excitation and ramp angle values are selected from relatively close range. However, data set patterns are placed far enough from each other to show diversity which can be observed by examining Figs. 9 and 13. In this way, the success rate of neural controllers changes in different directions on different data sets throughout the training process. Moreover, it is very common that neural controllers show changing success rates on each data set for different training configurations as shown in Tables 5, 7 and 8. In addition to these, trained neural controllers are capable of performing stable locomotion for speed excitation and ramp angle values which do not exist in any of these three data set. Thus, it is validated that generated three data sets carry necessary properties to avoid the over-fitting problem. One another important issue is the success differences between training, validation and testing data sets. In the generation of data sets, different intervals of speed excitation and ramp angle values were employed to be able to analyze behaviors of NNBC in larger input ranges. For this reason, successful locomotion percentages show diversity as explained in the data set preparation sub section. While the highest walking success is obtained in training data set, testing data set reaches the lowest success percentage due to enlarging input interval.

During NNBC driven biped locomotion experiments, the feedback taken from robot and environment are given to NNBCs so inputs diverge from training set inputs starting from the first control output which is different from CPG output. These dynamics may cause misunderstanding that NN training does not converge throughout the training process as seen in Fig. 9. In fact, NNBC successfully learns the relation between input and output data in the training set. Since walking simulations are performed with different inputs from inputs of data sets, first learning and then over-fitting occurs as shown in Fig. 9.

Even though NNs are highly non-linear structures, they reach a very high generalization performance in walking experiments. Especially, L_2 regularization contributed positively to walking success generalization. For instance, 30 of 32 inputs given to the neural controller are calculated during locomotion simulations in the torque control scenario but NNBC can interpret these previously unseen inputs to produce required torque values for stable walking. In a similar way, the neural controller reaches more than two times higher walking success for speed excitation and ramp angle combinations which are employed in data set generation at

Table 13 Results of CPG tuning experiments and new neural network based controller training trials

	Excitation Value Interval	Ramp Angle Interval	Improved CPG Walking Percentage	Torque Controller Neural Network Walking Percentage ($L_2 = 0.5$)	Position Controller Neural Network Walking Percentage ($L_2 = 0.05$)	PID Replacement Neural Network Walking Percentage ($L_2 = 0$)
Training	[3 10]	$[-10^\circ\ 10^\circ]$	31.22%	45.17%	55.94%	48.5%
Validation	[2 11]	$[-11^\circ\ 11^\circ]$	22.58%	30.42%	46%	40.75%
Testing	[1 12]	$[-12^\circ\ 12^\circ]$	16.67%	27.42%	43.01%	36.34%

the position control experiments. In addition to these, the generalization success of NNs is also validated with rough terrain, robot weight increase, and joint torque limitation experiments. Although neither CPG nor neural controller is trained in rough terrain environment, there are significant walking success differences between them in rough terrain environment and differences generally tend to increase with increasing roughness. In a similar way, neural controllers outperform CPG in robot weight experiments and success differences increase with increasing robot weight. It is interesting to see that proposed NNBCs become more successful in position control scenario than torque control scenario. The main reason for this difference may be related to the simplification of the legged locomotion problem by separating it into joint torque calculation and limb trajectory planning subparts similar to hierarchical control. Thanks to this separation, one NNBC interpolates limb angles while other focuses on torque generation for biped locomotion control. After that, the successful walking generation ability of the NNBCs is tested under the torque limitation constraint and NNBCs outperform the CPG again. Hence, we see that the proposed NNBCs can be optimized to work in small torque intervals and this ability may ease their usage in real-life biped locomotion control applications.

Remark 1 Different from the acquired generalization success with L_2 regularization, dropout which is another well-known regularization technique does not enhance neural controller performance for every control scenario. To measure performance increment that can be obtained with the dropout method, we retrain torque and position controller neural networks with their best training configurations by adding 0.1 and 0.2 dropout rates to the output of the LSTM layer. In our simulations, dropout does not increase the success rates for torque controller, but for position controller, some improvement is observed in certain cases. Obviously, this point deserves further investigation. This situation may be related to limited layer numbers in our proposed neural network architecture. We think that the dropout method would be more beneficial for the performance of neural networks if we would utilize a larger number of feedforward layers.

Due to the limited parameter space of CPG, it has limited adaptability to biped robot dynamics than NNBC. Even though Matsuoka oscillator is capable of sustaining oscillations, NNBC is more successful to evaluate the feedback taken from the robot model and environment. Moreover, it seems that LSTM layer in NNBC is capable of tracking phase changes of hybrid dynamics of biped robot platform thanks to internal structure of LSTM neuron model. Information which is carried by cell state can be easily modified or preserved with linear interactions between time steps. Moreover, nonexistence of nonlinear operation in the cell state line helps to avoid gradient vanishing type problems which may be seen in RNN training, see [73].

Remark 2 We perform further tuning on CPG parameters to test our assumption about the dependence of the generalization ability of neural networks on the tuning level of CPGs. Thus, we enlarge the walkable ramp angle range from $[-6.53, 6.53]$ to the $[-8.18, 9.55]$ degree interval for the training set walking patterns. That means an approximately 35.8% increase to the ramp angle range of CPG. A somewhat summary of our simulations by using the patterns generated by the improved CPG parameters are listed in Table 13. Here, we train torque controller, position controller, and PID replacement neural networks by using found best training configurations in previous experiments. All of them reaches higher success rates than CPG as expected. As can be seen, although the CPG success rates have been increased, so are the NN-based controller performances which was one of the main results of our work.

In the classical neural network-based controllers, feedforward layers are employed due to their quick training ability. However, feedforward layers do not have dynamic structure

so they may require several previous time step information from the controlled plant in the closed-loop control schemes. In addition, the number of required time steps may increase with nonlinearities in the plant. In our proposed structures, recurrent layer eliminates the need for previous time step information. We employed fully feedforward neural networks with 3 and 5 layers that have similar parameter counts with our proposed NNBC for position control scenarios as an ablation analysis work to see the importance of recurrent connections in proposed NNBC, see e.g. [74]. As shown in Table 4, fully feedforward neural networks could not succeed to provide successful legged locomotion as much as proposed NNBCs. As a result of this ablation analysis, we see that fully feedforward neural networks are not suitable for the same schemes that we used for proposed NNBC which includes a LSTM recurrent neural layer. We think that when previous time inputs are added to these feedforward controllers their success rates will increase but the number of the required time steps is unknown and it may increase for hybrid dynamic systems such as biped robots. On the other hand, the inclusion of previous time step information requires the placement of a higher number of network parameters to the input layer. So, the total neuron count will decrease in the feedforward neural network and this situation will also limit the performance of the controller. For these reasons, using the LSTM neuron model in the recurrent layer also seems efficient in terms of the number of parameters.

Note that LSTM neuron model is developed as a memory unit so its mathematical operation modeling capability is limited. When internal structure of LSTM neuron model is examined, it is seen that there is no derivative block to determine the rate of change at the error signal and integrator block to accumulate error signal. Different from the PID controller these mathematical operations have to be performed in between time steps by collaboration with other LSTM cells at the recurrent layer. To optimize this collaboration, a great number of network weight needs to be adjusted and this parameter adjustment may take long training duration. As a remedy to this weakness, closed loop control performance of the LSTM cell may be increased with internal structural modifications such as adding integrator and derivative gates.

5 Conclusion

In this study, we have focused on the NNBC design problem for two-legged robot motion control. We utilized LSTM neurons at the recurrent layer and linear feedforward neurons at the regression layer in the proposed neural controller architecture. We proposed different neural controller structures which generate either joint torques or limb angles to achieve stable walking. These neural controllers were trained with different training options. Then, their stable walking performances were evaluated and compared in a simulation environment.

As the main contribution of this work, we proposed NNBCs with LSTM recurrent layer instead of classical fully feedforward NNBCs for biped robot locomotion control and we demonstrated that a stable walking performance may be achieved in various walking environments. Note that biped robots have hybrid dynamics and as a result they exhibit different behavior during flight and stance phases for each leg. Since LSTM networks have certain memory, we expect that they might be able to track these changes and this property may contribute to increase the stable walking performance. Secondly, to support this idea proposed hybrid neural controllers were utilized in the feedback loop and feedforward paths. In this way, their robot dynamic change tracking ability of the recurrent layer in the NNBC was analyzed depending on controller placement. Thirdly, the performances of proposed controllers were validated in the simulation environment to show the robustness of the proposed structures under varying ground roughness conditions, robot weight changes, and joint torque limitations for position and torque control scenarios. Throughout these simulations, we showed that the proposed NNBCs perform better than CPG and PID type controller alternatives in the legged locomotion control problem. Fourthly, we benefit from well-known stability analysis methods COG, ZMP and limit cycle analysis to understand behavioral characteristic of proposed NNBCs as much as the robot model allows. As a final contribution, generalization abilities of proposed NNBCs were demonstrated and training properties that may affect generalization performance were investigated with walking simulations. As a result, L_2 regularization was found as the most important factor in the training of networks to reach higher walking success. Mini-batch sizes were determined as less effective but an important factor in generalization. Depending on the control scenario, the contribution of L_2 regularization showed diversity. In the replacement of the PID controller with a NNBC, L_2 regularization affected the performance of the controller negatively as compared to other trained neural controllers. To sum up, proposed NNBCs performed better for a wide range of ramp angles, walking speeds, rough terrain environments, robot weight changes, and joint torque limitations than their counterparts in terms of simulation results.

Finally, the propriety of data set generation, over-fitting issue and limitations of the LSTM neuron model are discussed and possible improvement ways are proposed related to neuron model and training options. The use of LSTM recurrent layer lets the neural controller detect phase changes between stance and flight without explicitly given foot contact information to the NN. Also, the stable walking conditions are widened by the generalization ability of RNNs.

Thus, the advantages of the RNN usage in the control of the hybrid dynamical systems are exemplified with a biped robot platform walking control problem.

In the near future, we will examine the generalization - abilities of the proposed controllers at varying ground stiffness. Subsequently, we will explore possible gains by widening the proposed neural controller with parallel neural layers rather than serial flow as in this study. In the long term, we plan to include robot platform dynamics into the NN training process by representing the biped robot with another NN.

Appendix A

A.1 Equations of Biped Robot Model

Following equations are taken from Taga et al. [27] and they are added to appendix for the sake of completeness. They are employed to generate data sets and test the locomotion control ability of trained neural networks in the paper. In addition to these, x_5, x_8, x_{11} and x_{14} are referred as θ_1, θ_2, θ_3 and θ_4 for the sake of simplicity throughout the paper, respectively. For further details see [27].

System Parameters

$$M = 48, m_1 = 7, m_2 = 4, l_1 = 0.5, l_2 = 0.6$$
$$I_1 = \frac{m_1 l_1^2}{12}, I_2 = \frac{m_2 l_2^2}{12}, b_1 = 10, b_2 = 10, g = 9.8,$$
$$b_k = 1000, k_k = 10000, k_g = 10000, b_g = 1000,$$

Initial Conditions

$$x_1 = 0.0, x_2 = 1.09, x_5 = 0.45\pi, x_8 = 0.57\pi,$$
$$x_3 = x_1 + \frac{l_1}{2}\cos x_5, x_4 = x_2 - \frac{l_1}{2}\sin x_5, x_{11} = 0.45\pi,$$
$$x_6 = x_1 + \frac{l_1}{2}\cos x_8, x_7 = x_2 - \frac{l_1}{2}\sin x_8, x_{14} = 0.57\pi,$$
$$x_9 = l_1\cos x_5 + \frac{l_2}{2}\cos x_{11}, x_{12} = l_1\cos x_8 + \frac{l_2}{2}\cos x_{14},$$
$$x_{10} = x_2 - l_1\sin x5 - \frac{l_2}{2}\sin x_{11},$$
$$x_{13} = x_2 - l_1\sin x8 - \frac{l_2}{2}\sin x_{14},$$
$$\dot{x}_i = 0, (i = 1, 2, .., 14), \dot{u}_i, \dot{v}_i = 0, (i = 1, 2, .., 12)$$

Equations of Kinematic Constraints

$$x_1 = x_3 - \frac{l_1}{2}\cos x_5 = x_6 - \frac{l_1}{2}\cos x_8$$
$$x_2 = x_4 - \frac{l_1}{2}\sin x_5 = x_7 - \frac{l_1}{2}\sin x_8$$
$$x_3 + \frac{l_1}{2}\cos x_5 = x_9 - \frac{l_2}{2}\cos x_{11}$$
$$x_4 - \frac{l_1}{2}\sin x_5 = x_{10} + \frac{l_2}{2}\sin x_{11}$$
$$x_6 + \frac{l_1}{2}\cos x_8 = x_{12} - \frac{l_2}{2}\cos x_{14}$$
$$x_7 + \frac{l_1}{2}\sin x_8 = x_{13} + \frac{l_2}{2}\sin x_{14}$$
$$(x_r, y_r) = (x_9 + \frac{l_2}{2}\cos x_{11}, x_{10} - \frac{l_2}{2}\sin x_{11})$$
$$(x_l, y_l) = (x_{12} + \frac{l_2}{2}\cos x_{14}, x_{13} - \frac{l_2}{2}\sin x_{14})$$

Feedback Pathway

$$a_1 = a_3 = a_4 = a_6 = a_8 = 1.5,\ a_2 = 1.0,\ a_5 = a_7 = 3.0$$
$$\begin{aligned} Feed_1 = -Feed_2 &= a_3(x_{11} - \pi/2)h(F_{g2}) + a_4 h(F_{g4}) \\ &\quad + a_1(x_5 - \pi/2) - a_2(x_8 - \pi/2) \\ Feed_3 = -Feed_4 &= a_3(x_{14} - \pi/2)h(F_{g4}) + a_4 h(F_{g2}) \\ &\quad + a_1(x_8 - \pi/2) - a_2(x_5 - \pi/2) \\ Feed_5 = -Feed_6 &= a_5(\pi/2 - x_{14})h(F_{g4}) \\ Feed_7 = -Feed_8 &= a_5(\pi/2 - x_{11})h(F_{g2}) \\ Feed_9 = -Feed_{10} &= (a_6(\pi/2 - x_{11}) - a_8\dot{x}_{11})h(F_{g2}) \\ &\quad + a_7(\pi/2 - x_{14})h(F_{g4}) \\ Feed_{11} = -Feed_{12} &= (a_6(\pi/2 - x_{14}) - a_8\dot{x}_{14})h(F_{g4}) \\ &\quad + a_7(\pi/2 - x_{11})h(F_{g2}) - \end{aligned}$$

Equations of Motion

$$M\ddot{x}_1 = F_1 + F_3, M\ddot{x}_2 = F_2 + F_4 - Mg$$

$$m_1\ddot{x}_3 = -F_1 + F_5, m_1\ddot{x}_4 = -F_2 + F_6 - m_1 g$$

$$\begin{aligned} I_1\ddot{x}_5 = &-F_1\tfrac{l_1}{2}\sin x_5 - F_2\tfrac{l_1}{2}\cos x_5 - F_5\tfrac{l_1}{2}\sin x_5 \\ &-F_6\tfrac{l_1}{2}\cos x_5 - b_1|x_5 - \tfrac{\pi}{2}|\dot{x}_5 - k_k h(x_5 - x_{11}) \\ &-(b_2 + b_k f(x_5 - x_{11}))(\dot{x}_5 - \dot{x}_{11}) + T_{r1} + T_{r3} \end{aligned}$$

$$\begin{aligned} I_1\ddot{x}_8 = &-F_3\tfrac{l_1}{2}\sin x_8 - F_4\tfrac{l_1}{2}\cos x_8 - F_7\tfrac{l_1}{2}\sin x_8 \\ &-F_8\tfrac{l_1}{2}\cos x_8 - b_1|x_8 - \tfrac{\pi}{2}|\dot{x}_8 - k_k h(x_8 - x_{14}) \\ &-(b_2 + b_k f(x_8 - x_{14}))(\dot{x}_8 - \dot{x}_{14}) + T_{r2} + T_{r4} \end{aligned}$$

$$\begin{aligned} I_2\ddot{x}_{11} = &-F_5\tfrac{l_2}{2}\sin x_{11} - F_6\tfrac{l_2}{2}\cos x_{11} - F_{g1}\tfrac{l_2}{2}\sin x_{11} \\ &-F_{g2}\tfrac{l_2}{2}\cos x_{11} + k_k h(x_5 - x_{11}) - T_{r3} - T_{r5} \\ &-(b_2 + b_k f(x_5 - x_{11}))(\dot{x}_{11} - \dot{x}_5) \end{aligned}$$

$$\begin{aligned} I_2\ddot{x}_{14} = &-F_7\tfrac{l_2}{2}\sin x_{14} - F_8\tfrac{l_2}{2}\cos x_{14} - F_{g3}\tfrac{l_2}{2}\sin x_{14} \\ &-F_{g4}\tfrac{l_2}{2}\cos x_{14} + k_k h(x_8 - x_{14}) - T_{r4} - T_{r6} \\ &-(b_2 + b_k f(x_8 - x_{14}))(\dot{x}_{14} - \dot{x}_8) \end{aligned}$$

$$f(x) = \max(0, x), h(x) = \begin{cases} 0 \text{ for } x \le 0 \\ 1 \text{ for } x > 0 \end{cases}$$

$$F_{g1} = \begin{cases} -k_g(x_r - x_{r0}) - b_g\dot{x}_r & \text{for } y_r - y_g(x_r) < 0 \\ 0 & otherwise \end{cases}$$

$$F_{g2} = \begin{cases} k_g(y_{r0} - y_r) - b_g f(-\dot{y}_r) & \text{for } y_r - y_g(x_r) < 0 \\ 0 & otherwise \end{cases}$$

$$F_{g3} = \begin{cases} -k_g(x_l - x_{l0}) - b_g\dot{x}_l) & \text{for } y_l - y_g(x_l) < 0 \\ 0 & otherwise \end{cases}$$

$$F_{g4} = \begin{cases} -k_g(y_l - y_{l0}) - b_g\dot{y}_l) & \text{for } y_l - y_g(x_l) < 0 \\ 0 & otherwise \end{cases}$$

Acknowledgements We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Quadro P6000 GPU used for this research.

Author Contributions Bahadır Çatalbaş: Conceptualization, Software, Data curation, Formal analysis, Investigation, Visualization, Writing - original draft, Writing - review editing. Ömer Morgül: Conceptualization, Methodology, Formal analysis, Resources, Writing - original draft, Writing - review editing, Supervision.

Funding This work was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) through project 120E104.

Code Availibility In this study, datasets are artificially generated in the simulation environment. After that, neural networks are trained and tested with these datasets. Codes that are used to generate datasets and trained neural networks will be published upon acceptance. Biped locomotion videos can be seen from https://figshare.com/s/77c93575fc72aa26f367 link.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Feng, S., Xinjilefu, X., Atkeson, C.G., Kim, J.: Optimization based controller design and implementation for the atlas robot in the darpa robotics challenge finals. In: 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), pp. 1028–1035. IEEE (2015)
2. Guizzo, E.: By leaps and bounds: an exclusive look at how Boston dynamics is redefining robot agility. IEEE Spectrum **56**(12), 34–39 (2019)
3. Holmes, P., Full, R.J., Koditschek, D., Guckenheimer, J.: The dynamics of legged locomotion: models, analyses, and challenges. SIAM Review **48**(2), 207–304 (2006). https://doi.org/10.1137/S0036144504445133
4. Uyanık, İ., Saranlı, U., Morgül, Ö.: Adaptive control of a spring-mass hopper. In: 2011 IEEE International Conference on Robotics and Automation, pp. 2138–2143. IEEE (2011). https://doi.org/10.1109/ICRA.2011.5979726
5. Chignoli, M., Kim, D., Stanger-Jones, E., Kim, S.: The MIT Humanoid Robot: Design, Motion Planning, and Control For Acrobatic Behaviors. arXiv:2104.09025 (2021)
6. Schwind, W.J.: Spring Loaded Inverted Pendulum Running: a Plant Model. Ph.D. thesis, University of Michigan, USA (1998). http://hdl.handle.net/2027.42/131537
7. Uyanık, I., Ankaralı, M.M., Cowan, N.J., Saranlı, U., Morgül, Ö.: Identification of a vertical hopping robot model via harmonic transfer functions. Transactions of the Institute of Measurement and Control **38**(5), 501–511 (2016). https://doi.org/10.1177/2F0142331215583327
8. Shih, C.L.: Ascending and descending stairs for a biped robot. IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans **29**(3), 255–268 (1999). https://doi.org/10.1109/3468.759271
9. Nishiwaki, K., Kagami, S., Kuniyoshi, Y., Inaba, M., Inoue, H.: Online generation of humanoid walking motion based on a fast generation method of motion pattern that follows desired zmp. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2684–2689 (2002). https://doi.org/10.1109/IRDS.2002.1041675
10. Goswami, A.: Foot rotation indicator (fri) point: A new gait planning tool to evaluate postural stability of biped robots. In: Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), vol. 1, pp. 47–52. IEEE (1999). https://doi.org/10.1109/ROBOT.1999.769929
11. Popovic, M.B., Goswami, A., Herr, H.: Ground reference points in legged locomotion: definitions, biological trajectories and control implications. The International Journal of Robotics Research **24**(12), 1013–1032 (2005). https://doi.org/10.1177/2F0278364905058363
12. Ankarali, M.M., Saranli, U.: Stride-to-stride energy regulation for robust self-stability of a torque-actuated dissipative spring-mass hopper. Chaos: An Interdisciplinary Journal of Nonlinear Science **20**(3), 033121 (2010). https://doi.org/10.1063/1.3486803
13. Kerimoğlu, D., Morgül, Ö., Saranli, U.: Stability and control of planar compass gait walking with series-elastic ankle actuation. Transactions of the Institute of Measurement and Control **39**(3), 312–323 (2017). https://doi.org/10.1177/2F0142331216663823

14. Spröwitz, A., Tuleu, A., Vespignani, M., Ajallooeian, M., Badri, E., Ijspeert, A.J.: Towards dynamic trot gait locomotion: design, control, and experiments with cheetah-cub, a compliant quadruped robot. The International Journal of Robotics Research **32**(8), 932–950 (2013). https://doi.org/10.1177/2F0278364913489205
15. Sproewitz, A., Moeckel, R., Maye, J., Ijspeert, A.J.: Learning to move in modular robots using central pattern generators and online optimization. The International Journal of Robotics Research **27**(3–4), 423–443 (2008). https://doi.org/10.1177/2F0278364907088401
16. Crespi, A., Ijspeert, A.J.: Online optimization of swimming and crawling in an amphibious snake robot. IEEE Transactions on Robotics **24**(1), 75–87 (2008). https://doi.org/10.1109/TRO.2008.915426
17. Aoi, S., Tsuchiya, K.: Stability analysis of a simple walking model driven by an oscillator with a phase reset using sensory feedback. IEEE Transactions on Robotics **22**(2), 391–397 (2006). https://doi.org/10.1109/TRO.2006.870671
18. Ijspeert, A.J.: Central pattern generators for locomotion control in animals and robots: a review. Neural Networks **21**(4), 642–653 (2008). https://doi.org/10.1016/j.neunet.2008.03.014
19. André, J., Teixeira, C., Santos, C.P., Costa, L.: Adapting biped locomotion to sloped environments. Journal of Intelligent & Robotic Systems **80**(3–4), 625–640 (2015). https://doi.org/10.1007/s10846-015-0196-0
20. Santos, C.P., Alves, N., Moreno, J.C.: Biped locomotion control through a biomimetic cpg-based controller. Journal of Intelligent & Robotic Systems **85**(1), 47–70 (2017). https://doi.org/10.1007/s10846-016-0407-3
21. Liu, C., Yang, J., An, K., Chen, Q.: Rhythmic-reflex hybrid adaptive walking control of biped robot. Journal of Intelligent & Robotic Systems **94**(3–4), 603–619 (2019). https://doi.org/10.1007/s10846-018-0889-2
22. Ijspeert, A.J., Kodjabachian, J.: Evolution and development of a central pattern generator for the swimming of a lamprey. Artificial Life **5**(3), 247–269 (1999). https://doi.org/10.1162/106454699568773
23. Nakamura, Y., Mori, T., Sato, M.A., Ishii, S.: Reinforcement learning for a biped robot based on a cpg-actor-critic method. Neural Networks **20**(6), 723–735 (2007). https://doi.org/10.1016/j.neunet.2007.01.002
24. Matsuoka, K.: Mechanisms of frequency and pattern control in the neural rhythm generators. Biological Cybernetics **56**(5), 345–353 (1987). https://doi.org/10.1007/BF00319514
25. Maeda, Y., Ito, A., Ito, H.: Central pattern generator and its learning via simultaneous perturbation method. In: The 2012 International Joint Conference on Neural Networks (IJCNN), pp. 1–6 (2012). https://doi.org/10.1109/IJCNN.2012.6252803
26. Park, C.S., Hong, Y.D., Kim, J.H.: Evolutionary-optimized central pattern generator for stable modifiable bipedal walking. IEEE/ASME Transactions on Mechatronics **19**(4), 1374–1383 (2013). https://doi.org/10.1109/TMECH.2013.2281193
27. Taga, G., Yamaguchi, Y., Shimizu, H.: Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. Biological Cybernetics **65**(3), 147–159 (1991). https://doi.org/10.1007/BF00198086
28. Lu, Q., Tian, J.: Research on walking gait of biped robot based on a modified cpg model. Mathematical Problems in Engineering **2015**, 9 (2015). https://doi.org/10.1155/2015/793208
29. Lewis, F.L., Jagannathan, S., Yesildirak, A.: Neural Network Control of Robot Manipulators and Non-linear Systems. CRC Press (1998)
30. Zurada, J.M.: Introduction to artificial neural systems, vol. 8. West Publishing Company St, Paul (1992)
31. Haykin, S.: Neural Networks and Learning Machines, 3/E. Pearson Education India (2010)
32. Pearlmutter, B.A.: Gradient calculations for dynamic recurrent neural networks: a survey. IEEE Transactions on Neural networks **6**(5), 1212–1228 (1995). https://doi.org/10.1109/72.410363
33. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
34. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, vol. 25, pp. 1097–1105 (2012). http://kr.nvidia.com/content/tesla/pdf/machine-learning/imagenet-classification-with-deep-convolutional-nn.pdf
35. Çatalbaş, B., Çatalbaş, B., Morgül, Ö.: Human activity recognition with different artificial neural network based classifiers. In: 2017 25th Signal Processing and Communications Applications Conference (SIU), pp. 1–4. IEEE (2017). https://doi.org/10.1109/SIU.2017.7960559
36. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7263–7271 (2017). https://openaccess.thecvf.com/content_cvpr_2017/papers/Redmon_YOLO9000_Better_Faster_CVPR_2017_paper.pdf
37. Mesnil, G., He, X., Deng, L., Bengio, Y.: Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In: Interspeech, pp. 3771–3775 (2013). https://www.isca-speech.org/archive/archive_papers/interspeech_2013/i13_3771.pdf
38. Sutskever, I., Martens, J., Hinton, G.E.: Generating text with recurrent neural networks. In: Proceedings of the 28th International Conference on Machine Learning, pp. 1017–1024 (2011). https://icml.cc/2011/papers/524_icmlpaper.pdf
39. Hénaff, P., Scesa, V., Ouezdou, F.B., Bruneau, O.: Real time implementation of ctrnn and bptt algorithm to learn on-line biped robot balance: experiments on the standing posture. Control Engineering Practice **19**(1), 89–99 (2011). https://doi.org/10.1016/j.conengprac.2010.10.002
40. Çatalbaş, B.: Recurrent Neural Network Learning with an Application to the Control of Legged Locomotion. Master's thesis, Bilkent University, Turkey (2015). http://hdl.handle.net/11693/30072
41. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015). https://doi.org/10.1038/nature14236
42. Kingma, D.P., Ba, J.: Adam: a Method for Stochastic Optimization. arXiv:1412.6980 (2014)
43. Çatalbaş, B., Morgül, Ö.: A new learning algorithm: sinadamax. In: 2019 27th Signal Processing and Communications Applications Conference (SIU), pp. 1–4. IEEE (2019). https://doi.org/10.1109/SIU.2019.8806259
44. Rostro-Gonzalez, H., Cerna-Garcia, P.A., Trejo-Caballero, G., Garcia-Capulin, C.H., Ibarra-Manzano, M.A., Avina-Cervantes, J.G., Torres-Huitzil, C.: A cpg system based on spiking neurons for hexapod robot locomotion. Neurocomputing **170**, 47–54 (2015). https://doi.org/10.1016/j.neucom.2015.03.090
45. Jaramillo-Avila, U., Rostro-Gonzalez, H., Camuñas-Mesa, L.A., Romero-Troncoso, Rd.J., Linares-Barranco, B.: An address event representation-based processing system for a biped robot. International Journal of Advanced Robotic Systems **13**(1), 39 (2016). https://doi.org/10.5772/2F62321
46. Guerra-Hernandez, E.I., Espinal, A., Batres-Mendoza, P., Garcia-Capulin, C.H., Romero-Troncoso, R.D.J., Rostro-Gonzalez, H.: A fpga-based neuromorphic locomotion system for multi-legged robots. IEEE Access **5**, 8301–8312 (2017). https://doi.org/10.1109/ACCESS.2017.2696985

47. Gutierrez-Galan, D., Dominguez-Morales, J.P., Perez-Peña, F., Jimenez-Fernandez, A., Linares-Barranco, A.: Neuropod: a real-time neuromorphic spiking cpg applied to robotics. Neurocomputing **381**, 10–19 (2020). https://doi.org/10.1016/j.neucom.2019.11.007
48. Wright, J., Jordanov, I.: Intelligent approaches in locomotion-a review. Journal of Intelligent & Robotic Systems **80**(2), 255–277 (2015). https://doi.org/10.1007/s10846-014-0149-z
49. Auddy, S., Magg, S., Wermter, S.: Hierarchical control for bipedal locomotion using central pattern generators and neural networks. In: 2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob), pp. 13–18. IEEE, Oslo, Norway (2019). https://doi.org/10.1109/DEVLRN.2019.8850683
50. Mandava, R.K., Vundavilli, P.R.: An adaptive pid control algorithm for the two-legged robot walking on a slope. Neural Computing and Applications **32**, 3407–3421 (2020). https://doi.org/10.1007/s00521-019-04326-2
51. Janczak, A.: Identification of Nonlinear Systems using Neural Networks and Polynomial Models: a Block-Oriented Approach, vol. 310. Springer Science & Business Media (2004)
52. Çatalbaş, B., Çatalbaş, B., Morgül, Ö.: Two-legged robot system identification with artificial neural networks. In: 2020 28th Signal Processing and Communications Applications Conference (SIU), pp. 1–4. IEEE (2020). https://doi.org/10.1109/SIU49456.2020.9302094
53. Choi, H., Crump, C., Duriez, C., Elmquist, A., Hager, G., Han, D., Hearl, F., Hodgins, J., Jain, A., Leve, F., et al.: On the use of simulation in robotics: opportunities, challenges, and suggestions for moving forward. Proceedings of the National Academy of Sciences **118**(1) (2021). https://doi.org/10.1073/pnas.1907856118
54. Olaru, A.D., Olaru, S.A., Mihai, N.F., Smidova, N.M.: Animation in robotics with labview instrumentation. International Journal of Modeling and Optimization **9**, 34–40 (2019). http://www.ijmo.org/vol9/680-RA05.pdf
55. Liu, C.K., Negrut, D.: The role of physics-based simulators in robotics. Annual Review of Control, Robotics, and Autonomous Systems **4** (2020). https://doi.org/10.1146/annurev-control-072220-093055
56. Rudenko, A., Palmieri, L., Herman, M., Kitani, K.M., Gavrila, D.M., Arras, K.O.: Human motion trajectory prediction: a survey. The International Journal of Robotics Research **39**(8), 895–935 (2020). https://doi.org/10.1177/0278364920917446
57. Xu, T., An, D., Jia, Y., Yue, Y.: A review: Point cloud-based 3d human joints estimation. Sensors **21**(5), 1684 (2021). https://doi.org/10.3390/s21051684
58. Bledt, G., Powell, M.J., Katz, B., Di Carlo, J., Wensing, P.M., Kim, S.: Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2245–2252. IEEE (2018). https://doi.org/10.1109/IROS.2018.8593885
59. Focchi, M., Del Prete, A., Havoutis, I., Featherstone, R., Caldwell, D.G., Semini, C.: High-slope terrain locomotion for torque-controlled quadruped robots. Autonomous Robots **41**(1), 259–272 (2017). https://doi.org/10.1007/s10514-016-9573-1
60. Nguyen, Q., Powell, M.J., Katz, B., Di Carlo, J., Kim, S.: Optimized jumping on the mit cheetah 3 robot. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 7448–7454. IEEE (2019). https://doi.org/10.1109/ICRA.2019.8794449
61. Li, J., Nguyen, Q.: Force-and-moment-based model predictive control for achieving highly dynamic locomotion on bipedal robots. arXiv:2104.00065 (2021)
62. Plestan, F., Grizzle, J.W., Westervelt, E.R., Abba, G.: Stable walking of a 7-DOF biped robot. IEEE Transactions on Robotics and Automation **19**(4), 653–668 (2003)
63. Vazquez, JA and Velasco-Villa, Martín: Numerical analysis of the sliding effects of a 5-DOF biped robot. In: 2011 8th International Conference on Electrical Engineering, Computing Science and Automatic Control, pp. 1–6. IEEE (2011)
64. You, Z., Zhang, Z.: An overview of the underactuated biped robots. In: 2011 IEEE International Conference on Information and Automation, pp. 772–776. IEEE (2011)
65. Barron-Zambrano, J.H., Torres-Huitzil, C.: Cpg implementations for robot locomotion: analysis and design. In: Robotic Systems-Applications, Control and Programming. IntechOpen (2012)
66. Efe, M.Ö.: Neural network assisted computationally simple pid control of a quadrotor uav. IEEE Transactions on Industrial Informatics **7**(2), 354–361 (2011). https://doi.org/10.1109/TII.2011.2123906
67. Olah, C.: Understanding lstm networks (2015). http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed 17 Nov 2019
68. Pineda, F.J.: Generalization of back-propagation to recurrent neural networks. Physical Review Letters **59**(19), 2229 (1987). https://doi.org/10.1103/PhysRevLett.59.2229
69. Gomez, A.: Backpropogating an lstm: a numerical example (2016). https://medium.com/@aidangomez/let-s-do-this-f9b699de31d9. Accessed 17 Nov 2019
70. Krogh, A., Hertz, J.A.: A simple weight decay can improve generalization. In: Advances in Neural Information Processing Systems, pp. 950–957 (1992). https://proceedings.neurips.cc/paper/1991/file/8eefcfdf5990e441f0fb6f3fad709e21-Paper.pdf
71. Hobbelen, D.G.E., Wisse, M.: Limit cycle walking. In: Humanoid Robots, Human-like Machines. IntechOpen (2007)
72. Hamzaçebi, H.: Analysis and Control of Periodic Gaits in Legged Robots. Ph.D. thesis, Bilkent University (2017)
73. Hochreiter, S.: The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **6**(02), 107–116 (1998). https://doi.org/10.1142/S0218488598000094
74. Meyes, R., Lu, M., de Puiseau, C.W., Meisen, T.: Ablation studies in artificial neural networks. arXiv:1901.08644 (2019)

Bahadır Çatalbaş is a PhD candidate in Electrical and Electronics Engineering Department at Bilkent University. He also received his BSc and MSc degrees from the same department in June 2013 and September 2015, respectively. He is working in developing novel deep learning methods and he also interests in legged locomotion applications of deep learning. He is the recipient of the Scientific and Technological Research Council of Turkey (TÜBİTAK) Graduate Scholarship.

Ömer Morgül was born in İstanbul, Turkey, in 1959. He received the B. Sc. and M. Sc. degrees from the Technical University of İstanbul, in 1980 and 1982, respectively, and the Ph. D. degree from University of California, Berkeley, in 1989, all in electrical engineering. Since 1989 he has been with the Department of Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey, where he is now a Professor. His research interests are in the area of dynamical systems and control theory, including robotics, nonlinear systems, chaotic systems, infinite dimensional systems and neural networks. He is the author, with Z. H. Luo and B. Z. Guo, of *Stability and Stabilization of Infinite Dimensional Systems with Applications* (1999).