Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 53-2 (2020) 1664–1670

Training Neural Networks for Plant Estimation, Control and Disturbance Rejection

Henry Kotzé **Herman Kamper** **Hendrik W. Jordaan**

Department of Electrical & Electronic Engineering, Stellenbosch University, South Africa, (e-mail: 19231865@sun.ac.za; kamperh@sun.ac.za; wjordaan@sun.ac.za).

Abstract: Neural networks are used in control systems to combat difficulties which nonlinear and linear controllers struggle to compensate for, such as environmental and model uncertainties. Neural networks have shown promising results as controllers or estimators of these uncertainties. However, few studies expand on important aspects on using and training a neural network, such as the dataset, input and output pairs, and the training of the different controllers and estimators. In this paper, a dataset used for neural controllers and estimators are presented which contains more complexity than that of the expected test environment. The training of different neural controllers and estimators are presented: estimators for the forward dynamics and disturbances, a feedback controller, a feedback linearisation controller and a disturbance rejection controller. For each neural component, the input and output pairs are presented with results of them performing in a test environment. From these results it was evident that through the use of the proposed dataset and training method the neural networks succeeded in fulfilling its role in the control architectures.

Keywords: Neural and fuzzy adaptive control, Adaptive observer design, Nonlinear adaptive control

1. INTRODUCTION

Linear controllers are excellent tools when operating on linear systems. However, these controllers are restricted to a small region when manipulating nonlinear systems. Nonlinear systems, even when derived from Newtonian or Lagrangian mechanics, are usually still far from the reality. They do not include saturation of motors, state dependent friction and stochastic processes influencing the system. The omitted behaviour results in deteriorated performance from the simulated responses, and requires continuous tuning of controller parameters in a changing environment. This calls for control architectures that are as easy to implement as the well-established linear controllers, but allows them to be more effective outside their designed region and adapt to a changing nonlinear environment or model.

Neural networks have the ability to approximate highly nonlinear functions and relearn them as they are changing. Given enough data and modelling capacity, neural networks will approximate the nonlinear and unmodelled behaviour of the plant and environment which control systems are operating within.

Linear controllers can also be augmented with neural networks to provide the adaptation to operate within a changing environment and accommodate the nonlinearity of the environment to increase its effectiveness. The role of the neural network is to assist the linear controller observe a more linear plant in a stochastic nonlinear environment. The augmentation of linear controllers by neural networks have shown to be successful in acting as various components in a control architecture.

Neural networks have acted as feedback linearisation controllers to accommodate for errors made by nonlinear controllers. This was implemented by Jiang et al. (2019) and Xiang et al. (2016) where the neural network is used to combat errors made by dynamic inversion controllers due to model uncertainties. They have also acted as the controllers to complex systems. Celen and Oniz (2018) and Al-Mahasneh et al. (2019) used neural networks to control rotary wing UAVs due to the ability of the neural networks to accommodate for unmodelled dynamics such as wind disturbances. The neural network's ability to estimate environmental disturbances and model uncertainty has shown promising results. Emran and Najjaran (2017) used a neural network to approximate the uncertainty in a model of a UAV such as mass and inertia variation, Allison et al. (2019) used a neural network to estimate the wind direction through the sensory data of a UAV, and Shi et al. (2019) used a neural network to estimate the ground effect when a UAV is flying close to the ground.

However, there are little discussion on the following points: how to generate the dataset on which these neural networks need to be trained on to fulfil its specific role in a control environment, the input and output pairs to be used for training, and the difference in training between the various roles. The main contribution of this work is to provide guidelines for generating a dataset for neural networks to act as controllers and observers in a control environment, showing the difference in training when act-

10.1016/j.ifacol.2020.12.2228

ing in their respective roles and providing the input and output pairs to be used during training for the various roles. The limited, but representative environment where a pendulum is controlled through an augmented linear controller was consider to act as the problem space for these neural networks.

Hwangbo et al. (2017), Koch et al. (2019) and Vankadari et al. (2018) used reinforcement learning to generate a controller. Within reinforcement learning, a neural network is trained to determine the correct control signal given the states of the system, but the paradigm and methodology are different from the training of classical neural networks which are presented in this work.

The problem space on which a linear controller will be operating is described followed by the design of the linear controller. A short description on neural networks is provided next, followed by how the dataset is generated. The last sections provide three neural-components which is used in a control environment.

2. PROBLEM SPACE

The plant being used throughout this paper is the pendulum system. It is chosen due to the fact that it has nonlinear behaviour similar to that of more complex systems, while being simple enough to keep the focus on the training of the neural network. It is common in many general complex robotics problems, and lays the foundation for moving towards more complex systems.

The pendulum system is described by

$$\ddot{\theta} + 2\zeta\omega_n\dot{\theta} + \omega_n^2 \sin(\theta) = 0, \tag{1}$$

with θ the angle of the pendulum, ω_n the natural frequency and ζ the viscous damping. These constants are chosen to have the values of 0.5 and 3 rad/s for ζ and ω_n respectively.

The augmentation of a control system with a neural network begins by designing a linear controller for the system. The pendulum system in (1) is first linearised at the stable equilibrium point, where θ is assumed to be small, with the dynamics of the system described by

$$\ddot{\theta} + 2\zeta\omega_n\dot{\theta} + \omega_n^2\theta = 0. \tag{2}$$

Using this linear model, a linear controller can be designed to control the linear system to a specification.

3. LINEAR CONTROLLER

The linear controller designed for the linear pendulum is a Proportional-Integral-Derivative (PID) controller, whose control signal is defined as

$$u = K_P(\theta_{ref} - \theta) + K_I \int_0^t (\theta_{ref} - \theta)dt + K_D \frac{d}{dt}(\theta_{ref} - \theta), \tag{3}$$

where θ_{ref} is the desired angular position of the pendulum and θ the angular position of the pendulum at the measured timestep. The gains are designed for a settling time of 3 s and a 10% percent overshoot. It results in a tuned PID controller with gains of $K_P = 4$, $K_I = 14$ and $K_D = 0.2$.

Operating within small angles, the linear controller will be capable of controlling the nonlinear model to specification. However, the further the system moves away from this region, the greater the effectiveness of the linear controller deteriorates. The introduction of neural networks in control systems is to increase this region of effectiveness.

4. TRAINING OF A NEURAL NETWORK

The feedforward neural network consists of a input layer, a hidden or multiple hidden layers and a output layer as shown in Fig. 1. Each layer contains units which performs a computational operation where one of these units can be seen on the left in Fig. 1 giving an exploded view. Each unit in a layer is connected to all of the units in the following layer, where each connection has its own weight, $W_{jk}^{(l)}$, and each unit has its own bias, $b_{jk}^{(l)}$. $W_{jk}^{(l)}$ is the weight between unit j of layer l and unit k of layer $l+1$. $b_j^{(l)}$ is the bias connected to unit j in layer l.

These units perform a computational operation taking multiple inputs and producing a scalar value. The computational operation sums all the incoming connections, adds the units bias value and then pushes this summation through a nonlinear activation function, σ:

$$x_{j;k+1} = \sigma(b_j + \sum_1^n w_{jk}^{(l)} x_j^{(l)}). \tag{4}$$

Before the unit can perform its computational operation the output of each unit in the previous layer is multiplied by their corresponding connections weight before arriving at the unit.

The weights and biases of the neural network are the trainable variables which are updated based on their effect on the loss function. The loss function used throughout this paper is the mean absolute error (MSE) with weight regularisation described by

$$L = \frac{1}{n}\sum_{i=0}^{n} \mid \hat{\boldsymbol{\theta}} - \boldsymbol{\theta} \mid + \lambda \sum_{l\in L}\sum_{j\in J}\sum_{k\in K} \mid W_{jk}^{(l)} \mid, \tag{5}$$

where n is the batch size, $\hat{\boldsymbol{\theta}}$ is the predicted output of the neural network, $\boldsymbol{\theta}$ is the ground truth vector, and λ is the weight regularisation coefficient. The effect of the trainable variables are calculated according to the gradient of the loss function with respect to each trainable variable:

$$\frac{\partial L}{\partial W_{jk}^{(l)}} = \frac{\partial L}{\partial \hat{\boldsymbol{\theta}}} \cdot \frac{\partial \hat{\boldsymbol{\theta}}}{\partial W_{jk}^{(l)}}. \tag{6}$$

Based on this gradient, the trainable variables are optimised using stochastic optimisers such as Adam (Kingma

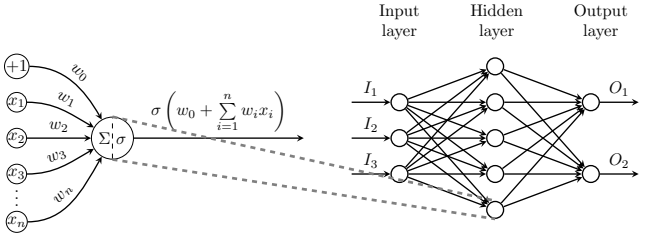

Fig. 1. Feedforward neural network with an exploded view of an unit (Veličković, 2016).

Table 1. Functions used to represent a control signal produced by a linear controller.

Function	Equation
w_1	$1(t-\tau_1)-1(t-\tau_1-\tau_2)$
w_2	$e^{\tau t}$
w_3	$m(t-\tau)$
w_4	at^2+bt+c

and Ba, 2014). These optimisers like Adam are used to provide better converging rates and lower loss values than classic stochastic gradient descent.

The neural networks used through this work is a standard feedforward neural network, with rectified linear (ReLU) activation functions for all units, except for the last layer which uses a identity activation function. The optimiser used is Adam.

5. GENERATING THE DATASET

The training of the various neural networks first occurs on synthetic data and should resemble the real environment in which the neural network will be operating. However, the synthetic data will not contain all the dynamics which exist within the real environment. The reason to use a synthetic dataset is to train the neural network to a good initial condition before training on real world data since real world data is scarce especially within robotics systems. A dataset for training neural networks to the initial condition before training on real world data will be presented here.

The training data is created by generating a random control input signal of the form

$$u(t)=\sum_{j=1}^{4} Y_j w(t,\tau_j), \tag{7}$$

where Y_j is a random magnitude selected from a uniform distribution and w_j is functions of the form in table 1.

The training data on which the neural network trains needs be sampled from the same distribution as that on which it will be tested to ensure good generalisation. The random control signal thus contains the expected properties from a signal produced by a linear controller which are step, ramp, exponential and square functions. Each of the parameters which describe these input functions are randomly selected from a uniform distribution over an appropriate range.

The generated control signal, $u(t)$, shown in Fig. 2 is propagated through the differential equation (1) and produces a random response, where the states and control input at each timestep are known as shown in Fig. 3. From this response, the input and output data can be selected.

The inputs and outputs are min-max normalised to combat issues of gradient vanishing. Furthermore, the training data are far more energetic and chaotic than that being produced by a linear controller, resulting in the training data containing more complexity than the test environment.

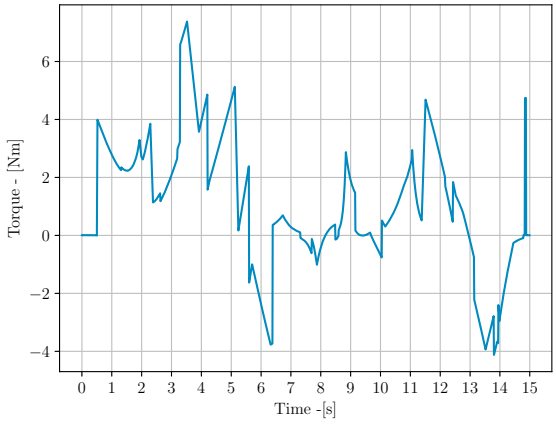

Fig. 2. A random control signal containing properties of a linear controller described by (7).

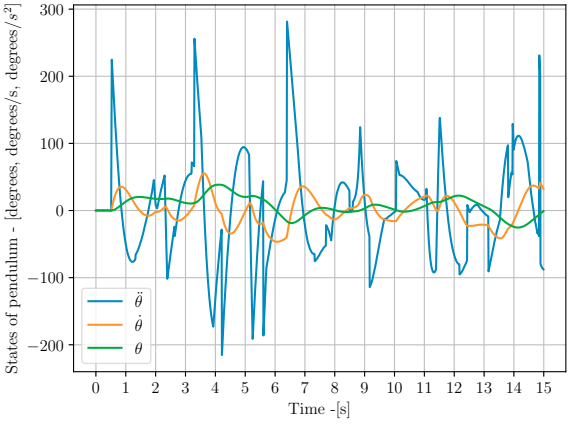

Fig. 3. Response of pendulum system from the control signal in Fig. 2.

6. NEURAL OBSERVER

The neural network will be used as its first task: as an estimator to the plant dynamics. The forward dynamics of the pendulum is represented by the state space representation of the system described by

$$\dot{\mathbf{x}}=\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\mathbf{f}(\boldsymbol{x},u)=\begin{bmatrix}x_2\\ -2\zeta\omega_n x_2-\omega_n^2\sin x_1+u\end{bmatrix}, \tag{8}$$

where $x_1=\theta$ and $x_2=\dot{\theta}$. From (8), the inputs and outputs of the neural network for the forward dynamics can be identified as the system states, $\boldsymbol{x}$, and the control input, u. The input and output pairs are summarised in table 2, where t is the timestep and N the time window or the number of timesteps the neural network sees in the past. The angular acceleration, $\ddot{\theta}$, is removed from the inputs to eliminate the behaviour of the neural network simply acting as a buffer: during training, $\ddot{\theta}[t+1]\approx\ddot{\theta}[t]$, causes the neural network to predict the correct state one timestep later and ignores all other inputs. This is unwanted behaviour, and the neural network would not aim to learn the dynamics.

Fig. 4 shows the response produced by (1) and a neural network representing the forward dynamics. The input to the system is an untrained input signal and it is evident that the trained neural network is a good approximation for the forward dynamics.

The neural network was trained on a relatively small dataset which contained 750k training and 150k validation samples. Since the training data is min-max normalised, the neural network is sensitive to any control signal greater than that on which it was trained. However, the magnitude of the control signal is known due to the hardware constraints of a physical system. Implementing these constraints in training will result in the neural network never seeing a control signal greater than in training.

7. INVERSE DYNAMICS CONTROLLER

Next, the training of a neural network behaving as a feedback controller positioned in the critical path is described.

The inverse dynamics of a system is described by the property

$$G^{-1}(s)G(s) = 1, \tag{9}$$

where $G(s)$ is the forward dynamics of the system and $G^{-1}(s)$ the inverse dynamics. The attractive property of the inverse dynamics when used as a controller results in the system being able to follow any reference signal. However, obtaining the inverse dynamics are difficult due to many systems being a many-to-one mapping from inputs to output and results in the existence of many solution for $G^{-1}(s)$.

A neural network is capable of learning the inverse dynamics and act as a controller to the plant by training on the correct input and output pairs. The inputs and outputs are described in table 3 which receives a time window of

Table 2. Input & output pairs for forward model neural network.

Inputs	Outputs
$\dot{\theta}[t:t-N]$	
$\theta[t:t-N]$	$\ddot{\theta}[t+1]$
$u[t:t-N]$	

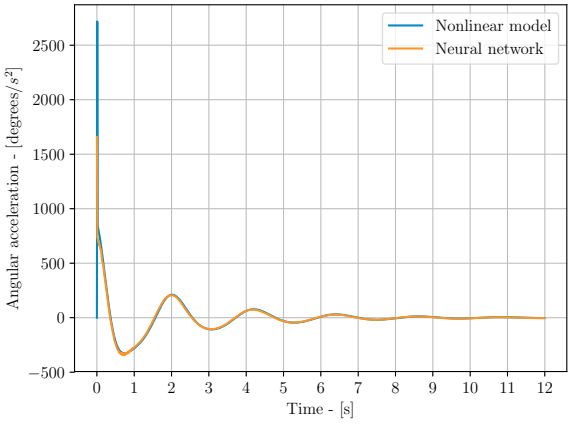

Fig. 4. Predicted response of a neural network representing the forward dynamics and response generated by (1). The reference angle was 90 degrees.

the pendulum's previous states and the desired states at the next timestep.

The neural network representing the inverse dynamics is used in the control architecture shown in Fig. 5. The purpose of the control architecture is to let the nonlinear plant behave as a linear plant. The control architecture begins by producing a control signal using the designed linear controller and propagating this through the linear model of the pendulum shown in (2). This provides the desired states for the inverse model to produce the control signal to set the nonlinear plant at the same states as that of the linear model.

Fig. 6 shows that the neural network has succeeded in producing the correct control signal to let the nonlinear plant behave as the linear model. Note that the nonlinear plant behaved linearly beyond the small angle assumption where the linear controller is effective and thus the augmentation increased the region of effectiveness.

As mentioned previously, due to min-max normalistion, the neural network will be sensitive to inputs greater than that on which it was trained. This is easily controllable by adding saturation blocks before the neural network inputs and since the training data contains more complexity, the neural network will be more robust since the test environment stays within the learned dynamics.

It can be concluded that training on the generated dataset, the trained neural neural network has learned the inverse dynamics and behaved as a controller to allow the nonlinear plant follow the linear model.

8. FEEDBACK LINEARISATION NEURAL NETWORK

The training of a neural network acting as a feedback controller with the goal of adding an addition control signal to linearise the nonlinear plant through online learning is presented next.

Neural networks acting as a feedback linearisation controllers are added to compensate for model uncertainties and errors made by nonlinear controllers. The architecture in Fig. 7 adds the neural network to feedback linearise the nonlinear plant and results in the linear controller observing a specified linear plant.

The neural network receives a window of the states of the chosen linear plant, nonlinear real plant and the linear control signal. From these inputs the neural network outputs a control signal which is summed with the linear controller

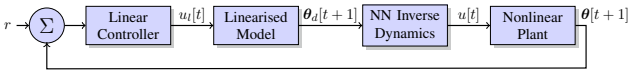

Fig. 5. Control architecture using the inverse model as the controller represented by a trained neural network.

Table 3. Input & output pairs for inverse model neural network.

Inputs	Outputs
$\ddot{\theta}_d[t+1]$	
$\ddot{\theta}[t:t-N]$	$u[t]$
$\theta[t:t-N]$	

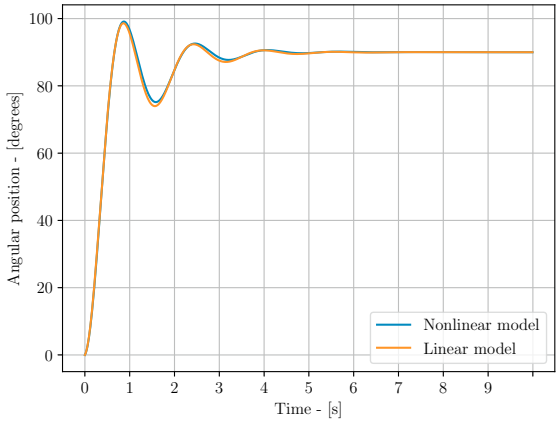

Fig. 6. Response of nonlinear model using neural controller trained on proposed dataset.

to accommodate for errors by the linear controller due to linearisation.

The correct control signal that the neural network should produce to feedback linearise the nonlinear plant is unknown given the inputs and thus a dataset with input and output pairs cannot be generated. This results in online learning of the neural network where the batch size, n, in (5) equals 1 and the trainable variables are optimised at each timestep.

Since the correct control signal being produced by the neural network is unknown at each timestep, it is only capable of knowing whether it produced the correct control signal by comparing the states of the linear and nonlinear model.

This results in the loss equation being a function of both the linear and nonlinear states of the plant describe by

$$L = \mid \boldsymbol{\theta} - \boldsymbol{\theta}_m \mid, \tag{10}$$

and not the output of the neural network, u_{nn}. The implication of the loss equation (10) for allowing backpropogation to occur is a function that relates the control signal to the states of the nonlinear plant. This is more clearly seen when the derivative of the loss function with respect to one of the weights are computed as described by

$$\frac{\partial L}{\partial W_{jk}^{(l)}} = \frac{\partial L}{\partial \theta} \cdot \frac{\partial \theta}{\partial u_{nn}} \cdot \frac{\partial u_{nn}}{\partial W_{jk}^{(l)}}. \tag{11}$$

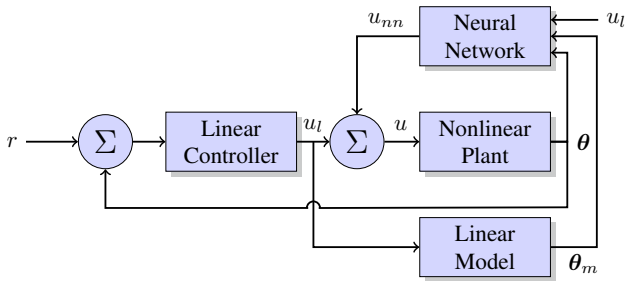

Fig. 7. A control architecture of an neural network acting as a feedback linearisation controller, causing the linear controller to observe a linear plant.

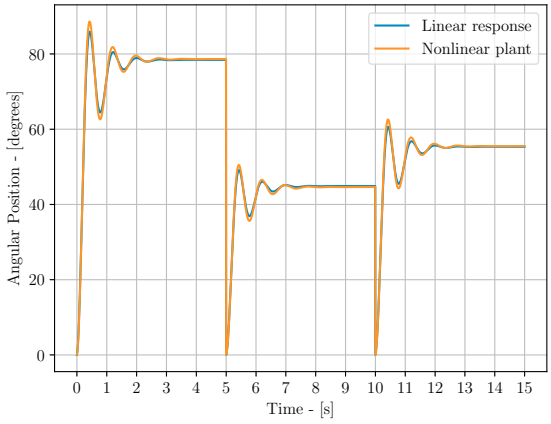

Fig. 8. Response of nonlinear plant being feedback linearised using a neural network.

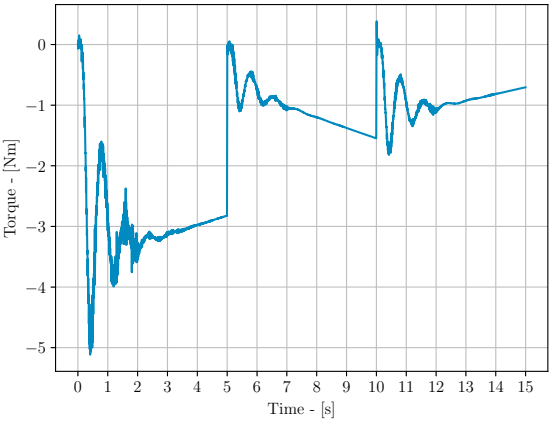

Fig. 9. Control signal produced by neural network acting as a feedback linearisation controller.

This is resolved by approximating the forward dynamics of the plant by a neural network, $\boldsymbol{\Theta}_{fd}$ and results in a loss function of

$$L = \mid \boldsymbol{\Theta}_{fd} - \boldsymbol{\theta}_m \mid . \tag{12}$$

This enables the gradient with respect to the trainable variables of the feedback neural network to be determined and results in

$$\frac{\partial L}{\partial W_{jk}^{(l)}} = \frac{\partial L}{\partial \boldsymbol{\Theta}_{fd}} \cdot \frac{\partial \boldsymbol{\Theta}_{fd}}{\partial u_{nn}} \cdot \frac{\partial u_{nn}}{\partial W_{jk}^{(l)}}. \tag{13}$$

It is assumed the forward dynamics of the plant has been previously learned and the weights and biases of the forward dynamics neural network are not adjusted during backpropagation. Approximating the forward dynamics by a neural network is owning to the fact that the mathematical equations derived from Newtonian or Lagrangian mechanics are usually still far from the reality and an effective feedback linearisation is dependent on the accuracy of the forward dynamics. The results presented in this section used (1) for allowing backpropagation to occur due to being in the simulated environment and describing the plant accurately.

Fig. 8 shows how the response of the pendulum system changes during the various steps. It is visible that the neural network has succeeded in adding the correct control signal to the linear controller and resulted in the system behaving linearly. The control signal being produced by the neural network is shown in Fig. 9.

The neural network started untrained during the response shown in Fig. 9 and initialised using Xavier initialisation (Glorot and Bengio, 2010). Stability during training is not guaranteed and there were cases when the system became unstable. Using this architecture on a physical system will require extensive simulated online training to ensure stability of the system.

From these results, online learning of a neural network acting as a feedback linearisation controller is feasible if an accurate forward model represented by a neural network is available to allow backpropagation to occur.

9. DISTURBANCE OBSERVER NEURAL NETWORK

The estimation of disturbances effecting a system have shown promising results and has been used in the feedback loop to reject its effect (Allison et al., 2019), (Shi et al., 2019). The training of such a neural network is used in the architecture shown in Fig. 10 and is used to estimate the disturbance effecting the pendulum system.

The training data is generated by propagating the random control signal of (7), through an undisturbed and disturbed plant, both being described by (1). The undisturbed plant would be estimated by a neural network. The disturbed plant's control signal is described by

$$u_{dp} = u_l + u_d \tag{14}$$

with u_l being the linear control signal and u_d representing the disturbance and is characterised by a random pulse train. The disturbance generates a different response from the undisturbed plant and produces the error from which the neural network needs to identify the disturbances as shown in Fig. 11.

The neural network receives a time window of the states of the disturbed plant, the undisturbed plant which is represented by a neural network and the control signal. Due to the difference in the response of the disturbed and undisturbed plant the neural network outputs an effective control signal that describes the disturbance influencing the system. The input and output pairs are summarised in table 4.

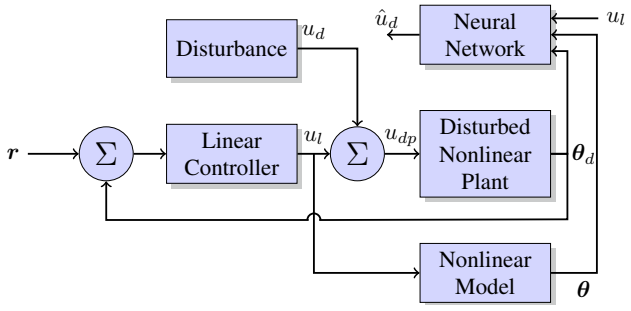

Fig. 10. The control architecture used for training a neural network to estimate the disturbance influencing the system.

Table 4. Input & output pairs for disturbance observer neural network.

Inputs	Outputs
$\ddot{\theta}_d[t:t-N]$	
$\theta_d[t:t-N]$	
$\ddot{\theta}[t:t-N]$	$\hat{u}_d[t]$
$\theta[t:t-N]$	
$u_l[t:t-N]$	

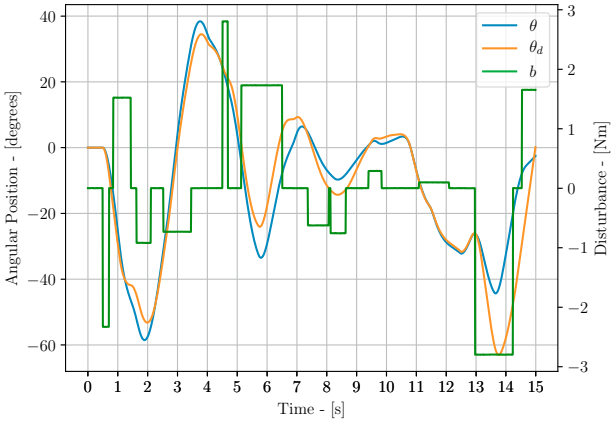

Fig. 11. Difference in response from a disturbed and undisturbed plant due to a random pulse train acting as a disturbance.

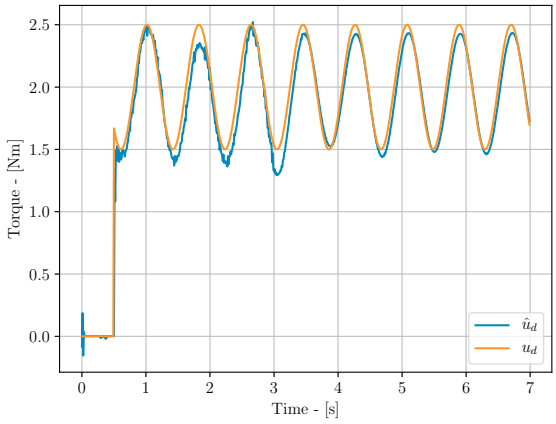

Fig. 12. Neural network identifying time varying disturbance.

Fig. 12 shows the neural network identifying a time varying disturbance influencing the pendulum system. The disturbance being identified is unseen due to the neural network being trained on random pulse train and shows the neural network generalises well.

The estimation of the disturbance can now be used in a feedback loop to reduce its effect. Fig. 13 shows the response when the estimated disturbance is subtracted from the control signal being received by the disturbed plant described by

$$u_{dp} = u_l + u_d - \hat{u}_d \tag{15}$$

and it is visible that the effect of the disturbance is reduced.

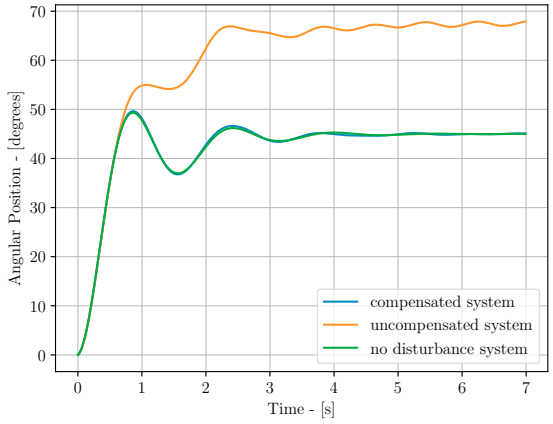

Fig. 13. Response of a disturbed -, undisturbed - and a disturbed plant whose disturbance is compensated by a neural network.

It can be concluded that using the generated dataset and superimposing a random pulse train as a disturbance, allowed the trained neural network to identify time varying disturbances in a control environment and successfully reject its effect.

10. CONCLUSION

Neural networks have been introduced in control systems to bring improvements where classical and nonlinear control strategies struggle to account for unmodelled environmental effects and model uncertainties. The neural networks have acted as various components in a control environment and shown promising results, however few expand on the training of these neural networks.

Thus, a dataset for training neural networks acting as controllers or estimators were presented. The dataset contained characteristic behaviour of a linear controller, but were more chaotic and energetic than that being produced by a linear controller. This results in the training data containing more complexity than the expected behaviour of a test environment.

The training of different neural controllers and estimators were presented. This includes feedback controllers, estimators and disturbance rejection controllers. For each controller or estimator the input and output pairs were presented and the difference in training of the different neural components were highlighted.

From the results, each neural-component, with the use of the generated dataset and the training method proposed allowed the neural-components to fulfil its role in the control architectures which are: estimating the plant dynamics, controlling the nonlinear plant to behave as a linear model, feedback linearise the nonlinear plant and estimating and rejecting disturbances.

REFERENCES

Al-Mahasneh, A.J., Anavatti, S.G., Ferdaus, M., and Garratt, M.A. (2019). Adaptive neural altitude control and attitude stabilization of a hexacopter with uncertain dynamics. In *2019 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, 44–49. IEEE.

Allison, S., Bai, H., and Jayaraman, B. (2019). Wind estimation using quadcopter motion: a machine learning approach.

Bari, S., Zehra Hamdani, S.S., Khan, H.U., ur Rehman, M., and Khan, H. (2019). Artificial neural network based self-tuned PID controller for flight control of quadcopter. In *2019 International Conference on Engineering and Emerging Technologies (ICEET)*, 1–5. IEEE.

Celen, B. and Oniz, Y. (2018). Trajectory tracking of a quadcopter using fuzzy logic and neural network controllers. In *2018 6th International Conference on Control Engineering & Information Technology (CEIT)*, 1–6. IEEE.

Emran, B.J. and Najjaran, H. (2017). Adaptive neural network control of quadrotor system under the presence of actuator constraints. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2619–2624. IEEE.

Furukawa, S., Kondo, S., Takanishi, A., and Lim, H.o. (2017). Radial basis function neural network based PID control for quad-rotor flying robot. In *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, 580–584. IEEE.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Y.W. Teh and M. Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, 249–256. PMLR, Chia Laguna Resort, Sardinia, Italy.

Hwangbo, J., Sa, I., Siegwart, R., and Hutter, M. (2017). Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, 2(4), 2096–2103.

Jiang, F., Pourpanah, F., and Hao, Q. (2019). Design, implementation and evaluation of a neural network based quadcopter UAV system. *IEEE Transactions on Industrial Electronics*, 1–1.

Kingma, D.P. and Ba, J. (2014). Adam: a method for stochastic optimization.

Koch, W., Mancuso, R., West, R., and Bestavros, A. (2019). Reinforcementl learning for UAV attitude control. *ACM Transactions on Cyber-Physical Systems*, 3(2), 1–21.

Shi, G., Shi, X., O'Connell, M., Yu, R., Azizzadenesheli, K., Anandkumar, A., Yue, Y., and Chung, S.J. (2019). Neural Lander: stable drone landing control using learned dynamics. In *2019 International Conference on Robotics and Automation (ICRA)*, 9784–9790. IEEE.

Vankadari, M.B., Das, K., Shinde, C., and Kumar, S. (2018). A reinforcement learning approach for autonomous control and landing of a quadrotor. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, 676–683. IEEE.

Veličković, P. (2016). Multilayer perceptron. Github.

Xiang, T., Jiang, F., Hao, Q., and Cong, W. (2016). Adaptive flight control for quadrotor UAVs with dynamic inversion and neural networks. In *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 174–179. IEEE.