*Article*

# Deep Learning Control for Digital Feedback Systems: Improved Performance with Robustness against Parameter Change

**Nuha A. S. Alwan [1] and Zahir M. Hussain [2,*]**

[1] College of Engineering, University of Baghdad, Baghdad, Iraq; n.alwan@ieee.org
ORCID: https://orcid.org/0000-0002-4040-9973
[2] Professor, School of Engineering, Edith Cowan University, Joondalup, Australia; z.hussain@ecu.edu.au
ORCID: https://orcid.org/0000-0002-1707-5485
* Correspondence: zmhussain@ieee.org ; z.hussain@ecu.edu.au

**Abstract:** Training data for a deep learning (DL) neural network (NN) controller are obtained from the input and output signals of a conventional digital controller that is designed to provide the suitable control signal to a specified plant within a feedback digital control system. It is found that if the DL controller is sufficiently deep (four hidden layers), it can outperform the conventional controller in terms of settling time of the system output transient response to a unit-step reference signal. That is, the DL controller introduces a damping effect. Moreover, it does not need to be retrained to operate with a reference signal of different magnitude, or under system parameter change. Such properties make the DL control more attractive for applications that may undergo parameter variation, like sensor networks.

**Keywords:** deep learning; feedback control; conventional controller; neural network; backpropagation.

## 1. Introduction

Design methods for feedback control systems are well-established. These include classical linear control system design, techniques for nonlinear control, robust control, H-∞ control and adaptive control. In addition, model-free control has emerged with techniques such as fuzzy logic control and artificial neural networks (ANN). The latter methods generally extend adaptive control techniques to nonlinear systems [1]. Closed-loop control applications of NNs are different from classification and image processing applications which are open-loop. NNs were first introduced in closed-loop control systems by Werbos in [2]. Offline learning in particular was formalized in [3] and was shown to yield important structural information. In addition, an important problem that also had to be addressed in closed-loop NN control was weight initialization for feedback stability [4]. In this work, a NN is trained and used for function approximation to replace a conventional controller in a digital feedback control system. Neural networks can model linear or nonlinear systems as they are excellent at finding the underlying processes that govern these systems. It has been stated in [1] that the 2-layer NN is sufficient for feedback control purposes. In the present work, however, we show that the addition of hidden layers, resulting in deep NN controllers, produces a damping effect and improves feedback control system stability. The multi-layer NN (specifically the 2-layer network with one hidden layer) took 30 years to effectively replace the single-layer NN which was introduced as early as the 1950s. The reason was the lack of the proper learning rule to update the hidden layer weights during training, a problem which was later solved by the backpropagation (BP) algorithm in 1986 [5]. The BP algorithm is also based on stochastic gradient descent (SGD) learning as in the single-layer NN, but uses a different method to update the gradient, namely, backpropagation. It took another 20 years to solve the poor performance issues of the deep NN (two or more hidden layers) through the innovation of deep learning (DL). DL, in essence, comprises many small technical improvements to ensure proper training. DL solves problems like vanishing gradient, overfitting and computational load [6, 7]. A deep NN with two hidden layers is shown in Figure 1. The inter-layer arrows in the figure represent weighted connections. It is a multiple-input, multiple-output (MIMO) system; however, in this work, only a single outlet is considered, with the hidden layers using a nonlinear activation function Φ(·), while the output layer uses a linear activation function Ψ(·). The output vector Y in Figure 1 can be expressed as follows:

$$Y_{L\times 1} = \Psi[M_{L\times K}\,\Phi\{H_{K\times J}\,\Phi(W_{J\times I}X_{I\times 1} + B_{J\times 1}) + \beta_{K\times 1}\} + b_{1\times 1}] \quad (1)$$

where X is the input vector, W, H, and M are the weight matrices of the first hidden, second hidden and output layers respectively, I, J, K, and L are the numbers of nodes of the input, first hidden, second hidden and output layers respectively. The variables B, β, and b represent the respective biases. The activation functions operate point-wise on the relevant vectors.

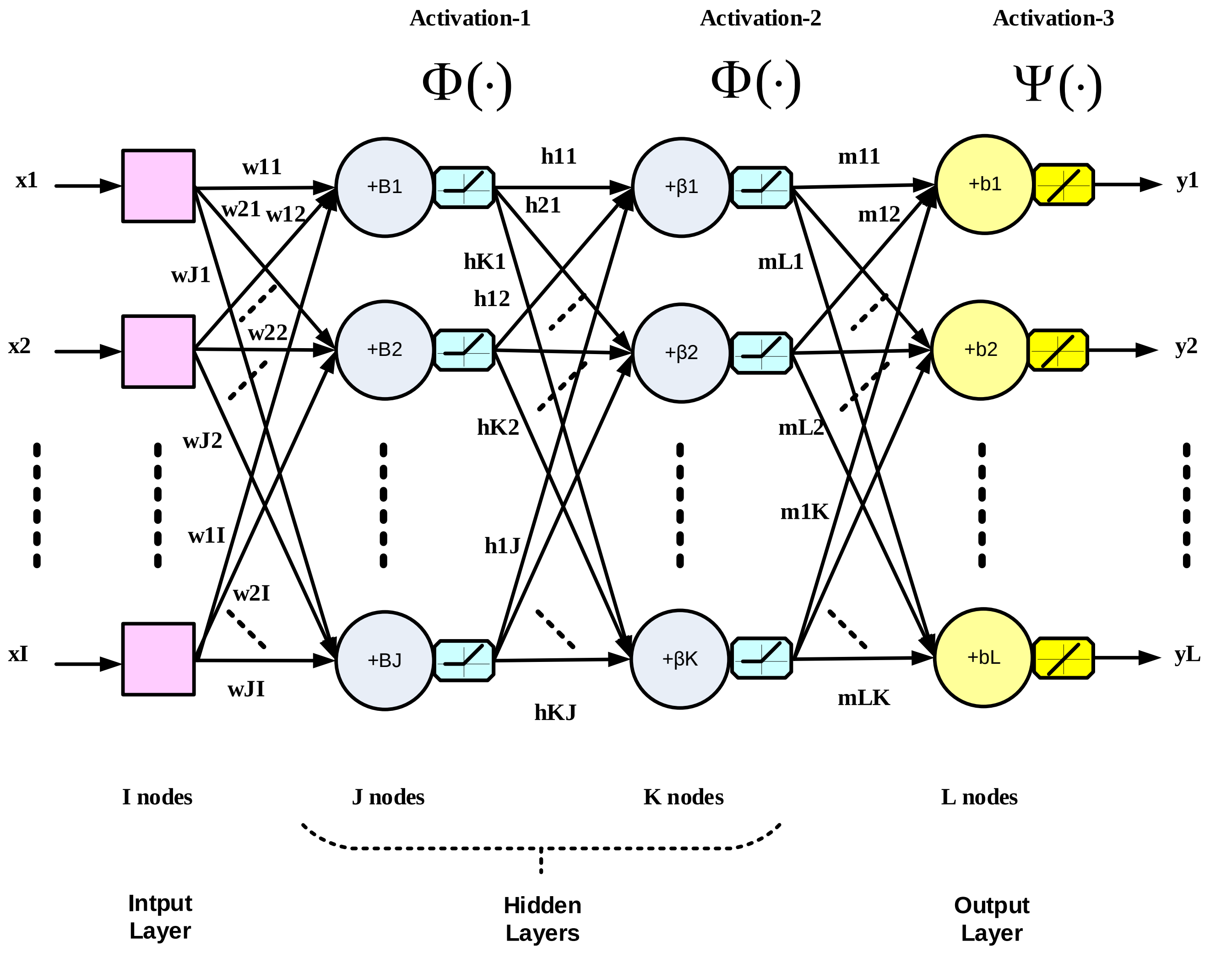

**Figure 1.** A generic diagram for a three-layer deep NN with two hidden layers.

In this work, a conventional controller is first designed and tested in a feedback control system. By using the input-output information of this controller as the training data of a learning algorithm such as the BP algorithm, a DL controller consisting of a deep NN is trained offline and then made to replace the conventional controller. Finally, the system or plant is controlled just by the DL controller and its performance monitored. It is shown that the feedback control system performance criteria can all be improved such as settling time, overshoot, steady-state error, etc.

Relevant works in the literature include [8] in which the speed of a DC motor is controlled in a feedback control loop with a proportional-integral-derivative (PID) controller which is the most commonly used in industry. Similar to the present work, DL is also resorted to in order to design an intelligent controller but via a deep belief network (DBN) algorithm [9]. A DBN performs a kind of unsupervised learning using a restricted Boltzmann machine (RBM) [9] to generate a set of initial weights to improve learning. RBM's are networks in which probabilistic states are learned for a set of inputs suitable for unsupervised learning. A similar approach to DL control is the work in [10], where RBMs are also used for weight initialization by unsupervised training. The disadvantage of DBNs is the hardware requirement since they consist of two stages, unsupervised pre-training and supervised fine tuning. The ordinary deep NNs used in the present work are less computationally demanding, and moreover, single-stage supervised offline learning is possible due to the availability of input-target data in the application considered.

In addition to supervised and unsupervised learning, there is also a third type of learning in machine learning called reinforcement learning (RL). This is learning by making and correcting mistakes in a trial and error fashion, that is, learning by experience in case of the absence of a training data set. It is a process in which a software agent makes observations and takes actions within an environment and in return, it receives rewards [11]. RL thereby achieves long-term results which are otherwise very difficult to achieve. Deep RL has recently been used in robotic manipulation controllers [12, 13]. However, RL is very computationally expensive and requires large amounts of data, and as such, it is not preferable to use to solve simple problems. RL suffers from the lack of real-world samples such as in robotic control where robotic hardware is expensive and undergoes wear and tear.

The rest of the paper is organized as follows: Section 2 presents the design of the conventional controller within the digital feedback control system. Section 3 explains the training procedure of the deep NN to construct the DL controller. The simulation results are given in Section 4, and, finally, Section 5 concludes the paper.

## 2. Conventional Controller Design for Digital Feedback Control System

The block diagram of a feedback control system with a digital controller $D(z)$ is shown in Figure 2. The z-transforms of the input sampled reference signal, the sampled error signal, the sampled control signal and the sampled output signal are denoted by $R(z)$, $E(z)$, $U(z)$ and $Y(z)$ respectively. We assume that $G(z)$ is the z-transform of $G(s)$ which is given by:

$$G(s)=G_o(s).G_p(s) \tag{2}$$

where $G_p(s)$ is the plant transfer function and $G_o(s)$ is the transfer function of the zero-order hold which represents a digital-to-analog converter that converts the sampled control signal to a continuous signal to be input to the plant. The zero-order hold takes the sample value and holds it constant for the duration of the sampling interval T. As such, $G_o(s)$ is given by [14]:

$$G_o(s)=\frac{1}{s}-\frac{1}{s}e^{-sT} \tag{3}$$

The closed loop transfer function is:

$$\frac{Y(z)}{R(z)}=T(z)=\frac{G(z)\,D(z)}{1+G(z)\,D(z)} \tag{4}$$

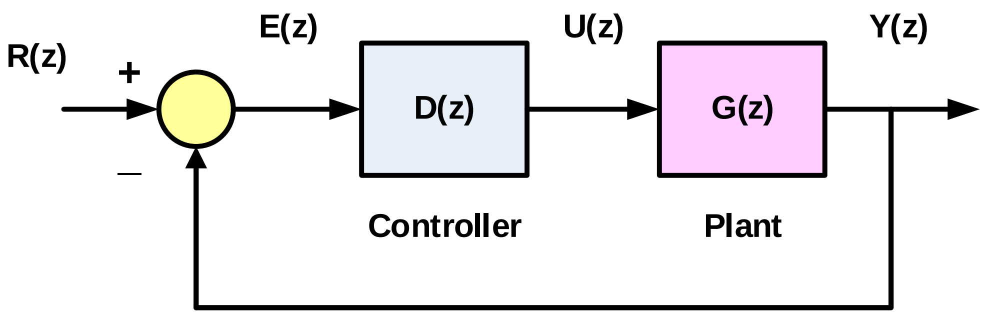

**Figure 2.** Block diagram of a feedback control system with a digital controller.

We will consider a second-order system with the plant transfer function given by:

$$G_p(s)=\frac{6960}{s(s+4)} \tag{5}$$

Note that, whereas a continuous second-order feedback control system is stable for all values of gain assuming left-half s-plane open-loop poles, a second-order sampled system can be unstable with increasing gain [14]. Using Equations (2), (3) and (5), we convert to digital as follows:

$$\begin{aligned} G(z) &= Z[G(s)] = Z[\frac{1-e^{-sT}}{s}\frac{6960}{s(s+4)}] \\ &= (1-z^{-1})Z[\frac{6960}{s^2(s+4)}] \\ &= 6960(1-z^{-1})Z[\frac{1}{4s^2}-\frac{1}{16s}+\frac{1}{16(s+4)}] \\ &= 1740(1-z^{-1})[T\frac{z}{(z-1)^2}-\frac{1}{4}\frac{z}{(z-1)}+\frac{1}{4}\frac{z}{(z-e^{-4T})} \end{aligned} \tag{6}$$

Substituting in the above for T by 0.001 sec, we obtain:

$$G(z)=\frac{0.003475z+0.003471}{z^2-1.996z+0.996} \tag{7}$$

An analog controller $G_c(s)$ can be designed such as to achieve a phase margin of 45◦ with a crossover frequency of 125 rad/sec. Using the compensation design methods in [14] for meeting phase margin specifications, we find that:

$$G_c(s)=\frac{5.6(s+50)}{s+312} \tag{8}$$

With T = 0.001sec, we find $D(z)=Z[G_c(s)]$ as:

$$D(z)=\frac{4.85z–4.61}{z-0.73} \tag{9}$$

Now the system in Figure 2 can be implemented in MATLAB with a unit-step reference input in order to obtain the digital error and control signals, e(n) and u(n)

doi:10.20944/preprints202104.0664.v1

respectively. These signals are shown in Figures 3 and 4, whereas the unit step response of the control system is shown in Figure 5. While these figures are simulation results, an analytical expression for the unit step response can be obtained. In Equation (4), substituting for $G(z)$ and $D(z)$ by Equations (7) and (9) respectively, and also substituting for the unit step reference input $R(z)$ by $z/(z-1)$, we solve by partial fractions to obtain:

$$Y(z)=\frac{z}{z-1}+\frac{1.9765z}{z-0.8979}+\frac{(-1.4798+j0.181)z}{z-(0.9056+j0.0862)}+\frac{(-1.4798-j0.181)z}{z-(0.9056-j0.0862)} \tag{10}$$

Taking the inverse Z-transform, we obtain the unit step response as:

$$y(n)=1+1.9765(0.8979)^n+2.9816(0.9096)^n\cos(0.0302\pi n+0.9615\pi);\quad n\geq 0 \tag{11}$$

Plotting the above expression coincides exactly with the plot of Figure 5. The conventional controller input-output signals are needed to train the DL controller described in the following section.

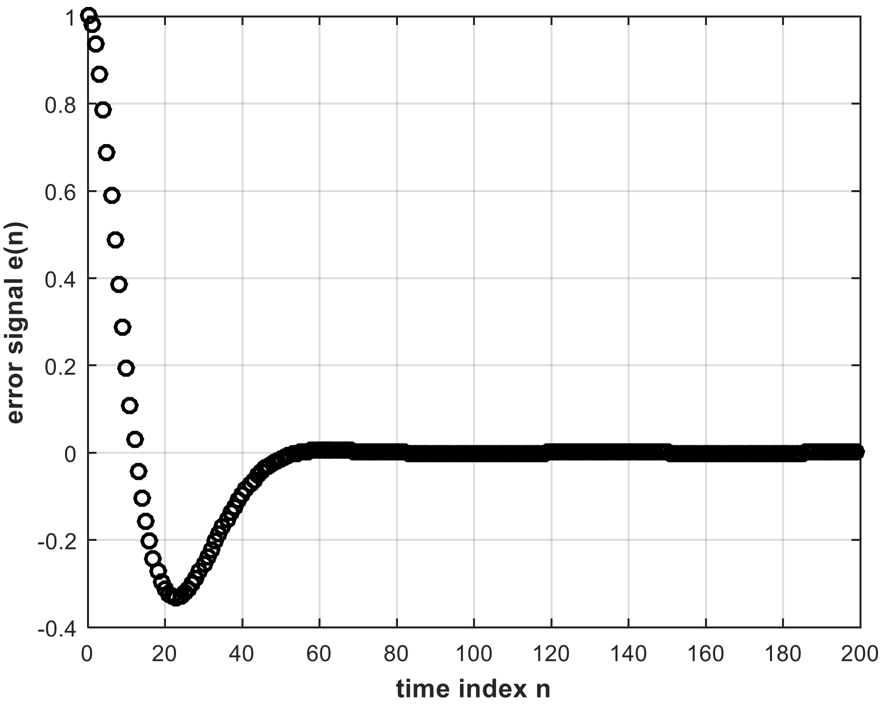

**Figure 3.** The error signal input to the digital controller.

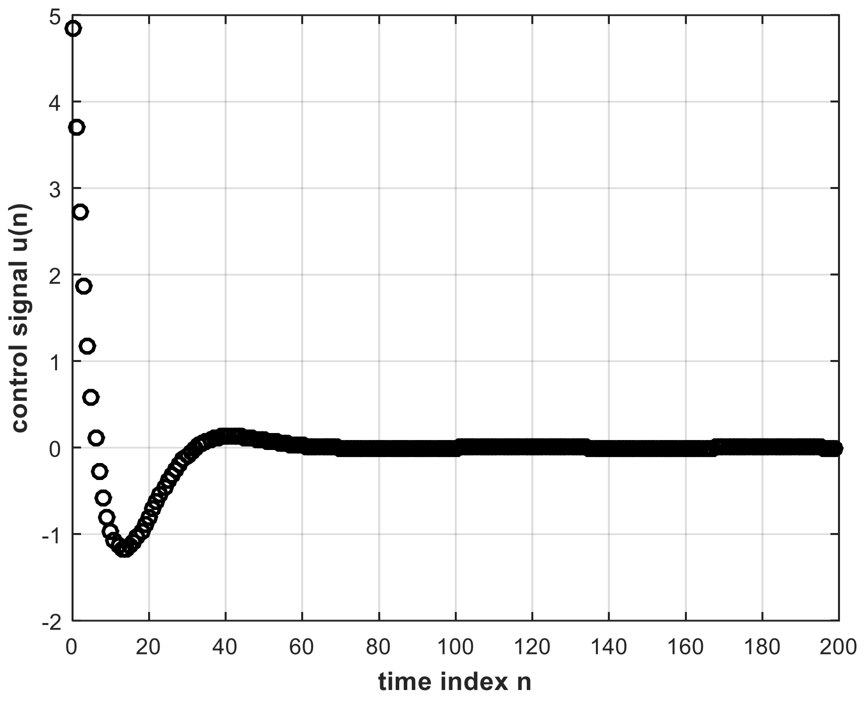

**Figure 4.** The control signal output from the digital controller.

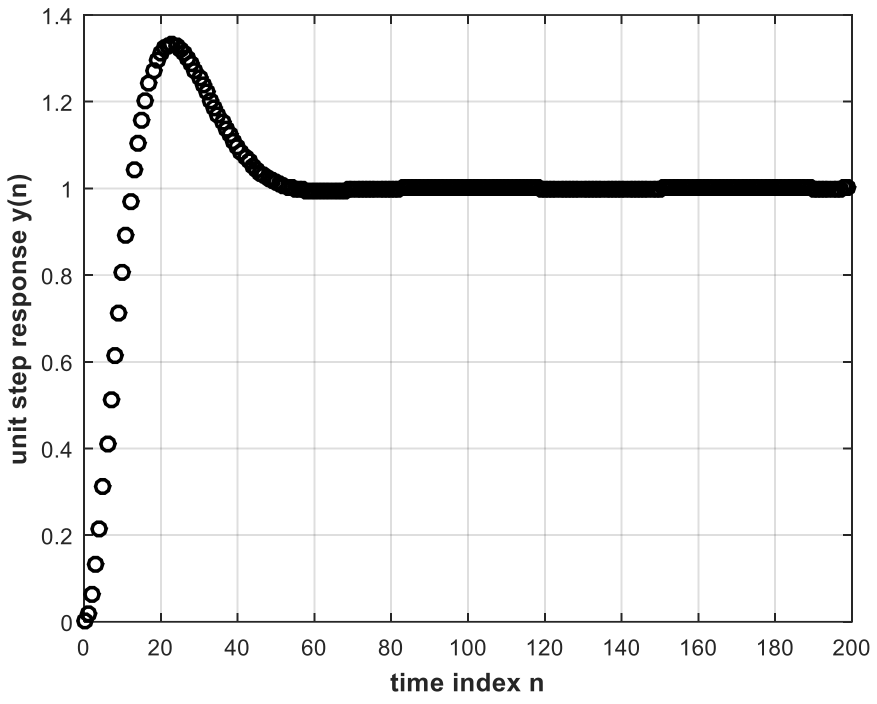

**Figure 5.** The unit step response of the feedback control system.

## 3. The DL Controller

The error and control signals of Figures 3 and 4 can be used to train a deep NN offline to obtain a DL version of the controller that learns the correspondence between e(n) and u(n). In other words, the training data consisting of input and correct output of the DL controller are the error input and control output signals of the conventional controller stored for a sufficient number of discrete time instants during feedback digital control

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 April 2021 doi:10.20944/preprints202104.0664.v1

system operation. Training is performed using the BP algorithm [6]. The trained DL controller is intended to replace the conventional controller for real-time feedback control system operation.

There are several training algorithms for training the neural network weights, the most important is the backpropagation (BP) algorithm, in which the output error starts from the output layer and propagates backwards until it reaches the hidden layer next to the input layer to update the weights. Based on the update strategy, there are different variations of BP, the most common is BP based on gradient descent (GD) [15]. With reference to Figure 1, the GD-based BP algorithm for updating the weights can be summarized by the following equations assuming a linear activation function for the output layer and nonlinear activation functions for the hidden layers.

$$
\begin{aligned}
& m_{lk} \leftarrow m_{lk} + \Delta m_{lk} \quad , \quad l = 1,\ldots..,L \ \ and \ \ k = 1,\ldots.,K. \\
& \text{where } \ \Delta m_{lk} = \alpha\,(d_l - y_l)\,y_k = \alpha\,\delta_l\,y_k \\
& h_{kj} \leftarrow h_{kj} + \Delta h_{kj} \quad , \quad k = 1,\ldots..,K \ \ and \ \ j = 1,\ldots.,J. \\
& \text{where } \ \Delta h_{kj} = \alpha\,\delta_k\,y_j \ \text{ with } \ \delta_k = [\sum_l m_{lk}\,\delta_l\,].\Phi'(v_k) \qquad (12) \\
& w_{ji} \leftarrow w_{ji} + \Delta w_{ji} \quad , \quad j = 1,\ldots..,J \ \ and \ \ i = 1,\ldots.,I. \\
& \text{where } \ \Delta w_{ji} = \alpha\,\delta_j\,x_i \ \text{ with } \ \delta_j = [\sum_k h_{kj}\,\delta_k\,].\Phi'(v_j)
\end{aligned}
$$

and so on for more hidden layers. In Equations (12) above, $\alpha$ is the BP learning rate, the *d's* are the correct outputs at the output layer needed for supervised training, the *v's* are the activation function inputs, the *y's* are the activation function outputs for output and hidden layers, the *x's* are the inputs to the input layer, and $\Phi'(.)$ is the derivative of the nonlinear activation function $\Phi(.)$.

In the present application, the training data consisting of the correct outputs (*d's*) and inputs (*x's*) would be values of the control signal u(n) and the error signal e(n) respectively.

It should be noted, however, that the correspondence between e(n) and u(n) is not one-to-one as can be seen from Figures 3 and 4. Instances can be found of a single value of e(n), occurring at different time indices, that corresponds to more than one value of u(n). These identical values of e(n) at different time indices can be made distinguishable from each other if their past values are taken into account. Error values may be the same for different time instants, but their past behaviors are normally different. Therefore, the input layer of the deep NN may consist of more than one node to take into consideration present as well as past values of the error signal, but the output layer has only one node such that the NN performs regression or function approximation. The activation functions of the hidden-layer nodes are chosen as the rectified linear unit (ReLU) function which is known to perform better with DL than the sigmoid function [6, 7]. The output layer single node, however, can be chosen to have a linear activation function. A neural network with two or more hidden layers is considered deep [7]. It is important to properly choose the convenient topology of a NN. Clearly, the number of neurons in the input layer is equal to the number of inputs, and the number of neurons in the output layer is likewise equal to the number of outputs. As for the number of neurons in the hidden layers, this is a problem facing many researchers. Rules of thumb are given in many instances in the literature [16, 17]. Some of these rules state that the number of nodes in the hidden layer should lie between those in the input and output layers, and the number of hidden nodes could be taken as two-thirds the sum of input and output layer nodes. In [18], it is concluded that the power of the NN does not depend on whether the first hidden layer or second hidden layer has more neurons, whereas [19] presents an empirical study of a 3-layer (two hidden layers) NN concluding that:

$$
n_1 = \lceil 0.5 n_h + 1 \rceil \ \ and \ \ n_2 = n_h - n_1 \qquad (13)
$$

where $n_1$ and $n_2$ are numbers of neurons in the first hidden and second hidden layers respectively, $\lceil\cdot\rceil$ is the ceil integer function, and $n_h$ is the total number of hidden neurons in both hidden layers. The trial-and-error approach, however, is often used to successfully determine the number of layers and the number of neurons in the hidden layers [18].

## 4. Simulation Results and Discussion

All simulations are implemented in MATLAB. Figures 3 and 4 demonstrate the training data that are used in offline training of the deep NN that constitutes our DL controller. The training data are the necessary input-correct output pairs needed for implementing the BP algorithm which is a supervised learning algorithm. The learning rate of the BP algorithm is taken by trial and error as 0.02. In each training or learning iteration, seven past samples plus the present sample of the error signal are used as input to the deep NN as discussed in Section 3. Thus, the input layer will consist of eight nodes. By trial and error, this number of necessary input nodes was found to yield optimum performance for this case of two hidden layers. The output layer will have only one node as only one controller output is required to provide the control signal to the plant. The output node is made to operate with a linear activation function. The number of neurons in each of the hidden layers is taken as 5. For two hidden layers, this number is almost in conformity with the empirical rule of Equation (13). The nonlinear activation function for all hidden nodes is the ReLU function which has the merit of solving the vanishing gradient problem for deep NNs trained by the BP algorithm [6, 7]. As in Figures 3 and 4, the number of available training data pairs is 200. Training for 200 iterations, called an epoch [7], is repeated 200 times. That is, the number of epochs is also 200. The DL controller with two hidden layers is first trained. Its weights are initialized with real random numbers between 1 and -1 taken from a uniform random distribution. However, the learning behavior and subsequent closed-loop stability was found to be sensitive to the initial values of the weights. Therefore, different initial random weights from the same uniform distribution were tried for good performance. The offline nature of our training procedure renders this possible. There exist, however, various weight initialization methods and algorithms in the literature [20, 21]. These can prove especially helpful in applications where the controller parameters are to be adjusted online [22]. When learning was achieved, the DL controller was used in inference mode to test its performance. In this mode, the same stored error signal from the conventional controller is entered to the trained DL controller, every sample with its seven past values, and the DL controller output is obtained as shown in Figure 6. It is clear that good learning is achieved as Figure 6 is almost identical to Figure 4.

The next step is to use the trained DL controller in real time to provide the control signal to the plant within the feedback control system. The unit step response is shown in Figure 7. This, in turn, is almost identical to Figure 5 that corresponds to the conventional controller whose behavior was learned.

The same trained DL controller is made to operate within the feedback system but subject to a step reference of magnitude 2. The resulting step response of the system is shown in Figure 8, where it is clear that the system yields the correct response with the same trained DL controller regardless of the step magnitude. Note that the input to the DL controller is different from the training input. That is, the DL controller does not exhibit any signs of overfitting [6]. The latter is a situation where the deep NN fails to respond correctly to other than the training data. That is why, in this work, we do not need to employ dropout which is another DL technique used to overcome overfitting [6, 7].

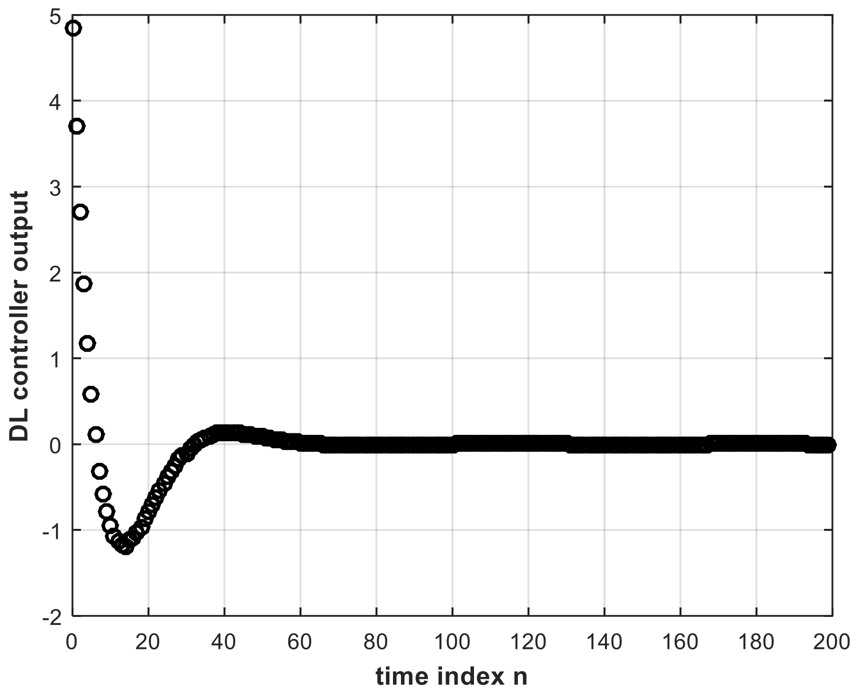

**Figure 6.** Output signal of trained DL controller (with two hidden layers) in inference mode.

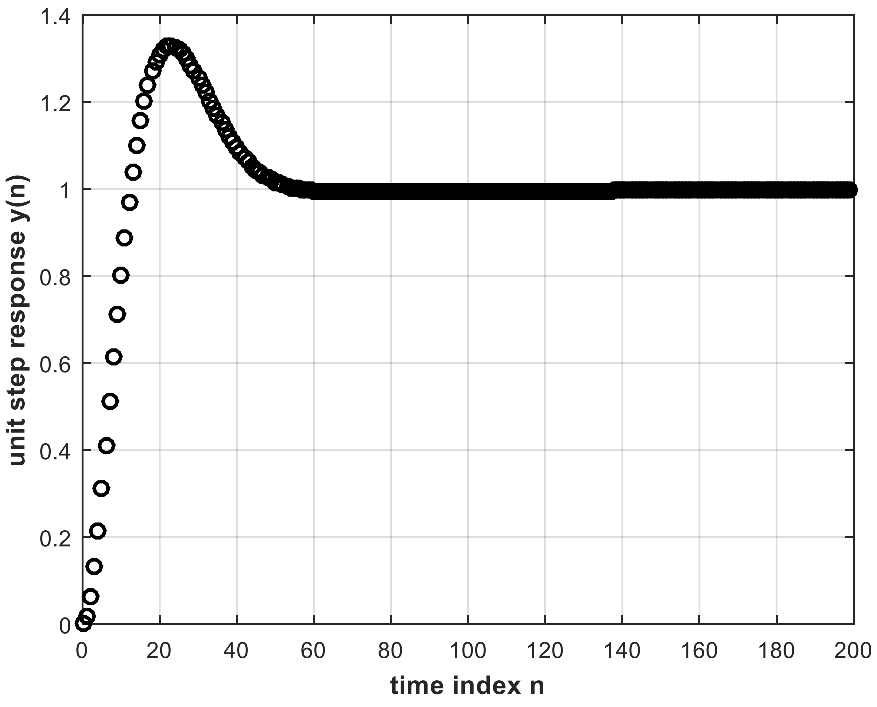

**Figure 7.** Unit step response of feedback control system using the trained DL controller with two hidden layers.

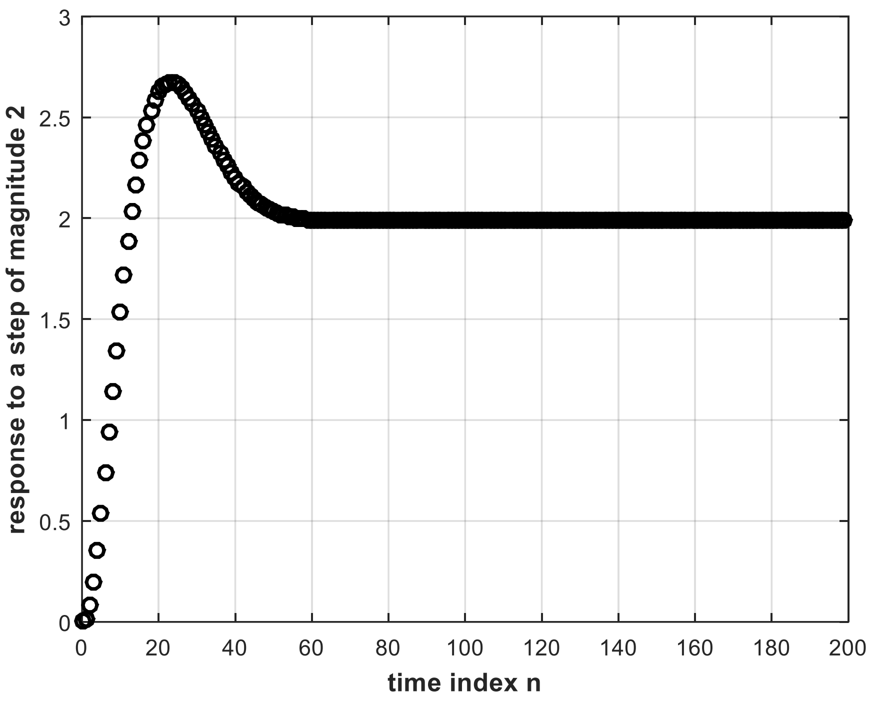

**Figure 8.** System response to a step magnitude of 2 using the trained DL controller.

We now investigate the use of deeper NNs for DL controllers in feedback control and compare between them as well as with the conventional controller in terms of settling time. The settling time of the second-order system under consideration is taken as the time needed for the system step response to reach a value within a certain percentage of the final value. The percentages considered are 2, 0.2 and 0.02. Table (1) lists the settling time of the different tested cases. The number of nodes is 5 in each hidden layer, and the number of input layer nodes is 8 for all cases. It can be seen from Table (1) that the deeper the controller the smaller (better) the settling time of the feedback control system. The reason is that better learning is achieved with deeper NNs. Moreover, with four hidden layers, the DL controller even outperforms the conventional controller in terms of settling time. So the DL controller has the advantage of producing a damping effect when sufficiently deep, thereby enhancing performance. The number of input layer nodes was fixed at 8 for all entries of Table (1) for the purpose of comparison. However, with four hidden layers, results are better if we set this number to 4. For this case, the feedback system unit step responses with the conventional and trained DL controllers are shown in Figure 9.

**Table 1.** Unit step response settling times for the different types of controllers under consideration.

| **Controller type** | **Settling times (milliseconds) within r % of final value** | | |
|---|---|---|---|
| | r = 2 | r = 0.2 | r = 0.02 |
| Conventional | 48 | 74 | 104 |
| DL with 2 hidden layers | 49 | > 200 | > 200 |
| DL with 3 hidden layers | 49 | 85 | 120 |
| DL with 4 hidden layers | 47 | 55 | 92 |

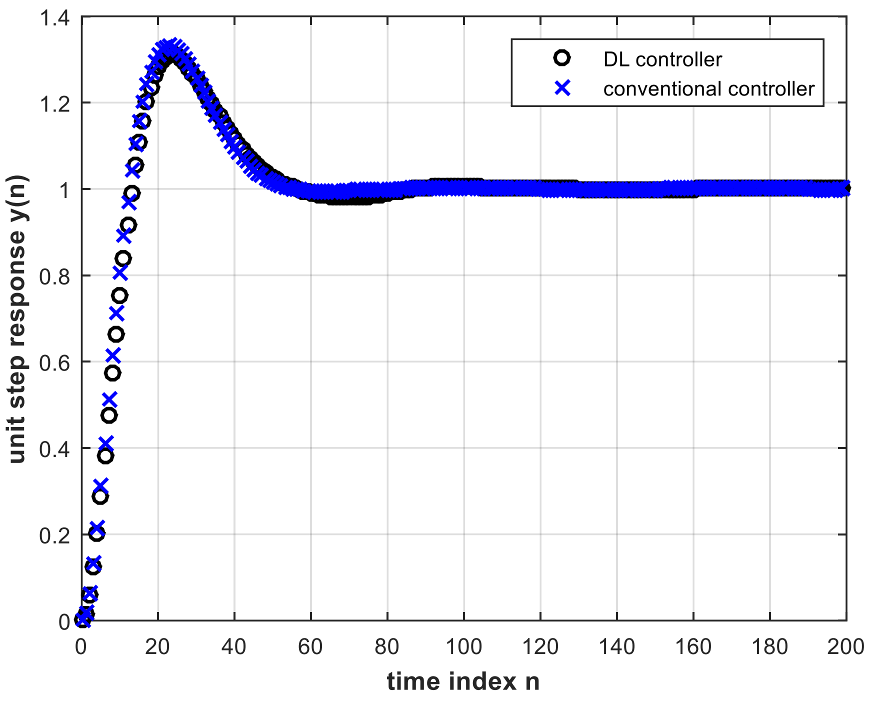

**Figure 9.** Unit step response of the feedback system with conventional and trained DL controllers. The number of hidden layers is 4 and the number of input nodes is 4.

Now we can study the influence of system parameter change on the performance of the trained DL controller. Let us assume that the plant gain in Equation (5) undergoes a considerable change from 6960 to 20000. Using Equation (2) and converting to digital with T = 0.001sec, we arrive at the following expression for G(z) using the same steps that led to Equation (7):

$$G(z)=\frac{0.009987\,z+0.009973}{z^2-1.996\,z+0.996} \qquad (14)$$

The trained DL controller for Figure 9 with four hidden layers and four input nodes is tested in online operation of the feedback control system with the above plant parameter change. The unit-step response is shown in Figure 10 together with the simulated conventional controller case (without re-design) under the same system parameter change.

Proceeding with a workout similar to that which led to Equation (11), the analytical expression for the feedback system unit step response using the conventional controller of Equation (9) under system parameter change is:

$$y(n)=0.9999+0.1433(0.9439)^n+1.0948(0.9050)^n\cos(0.0927\pi n+0.9936\pi);\; n\geq 0 \quad (15)$$

Plotting Equation (15) above coincides exactly with the corresponding plot in Figure 10. It is clear from Figure 10 that the DL controller behaves much more satisfactorily than the conventional controller. This means that the latter has to be redesigned for successful operation, whereas the DL controller does not need re-training. A magnified view of Figure 10 is shown in Figure 11.

The advantage of obtaining smaller settling time with the DL controller is evident. DL causes the system to reach steady state faster with less overshoot; the first-peak ratio is almost 8. Table (2) shows the settling times of the control system for both controllers. The steady state error for all above results is zero.

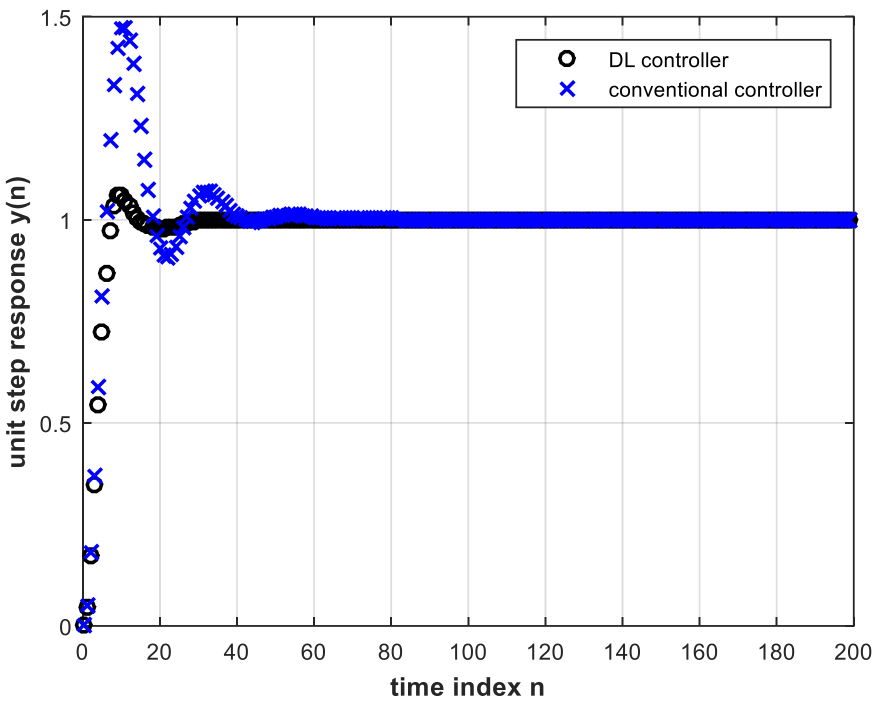

**Figure 10.** Unit step response of the feedback control system with the trained DL controller (four hidden layers and four input nodes) as well as with the conventional controller, both under system parameter change, Equation (14).

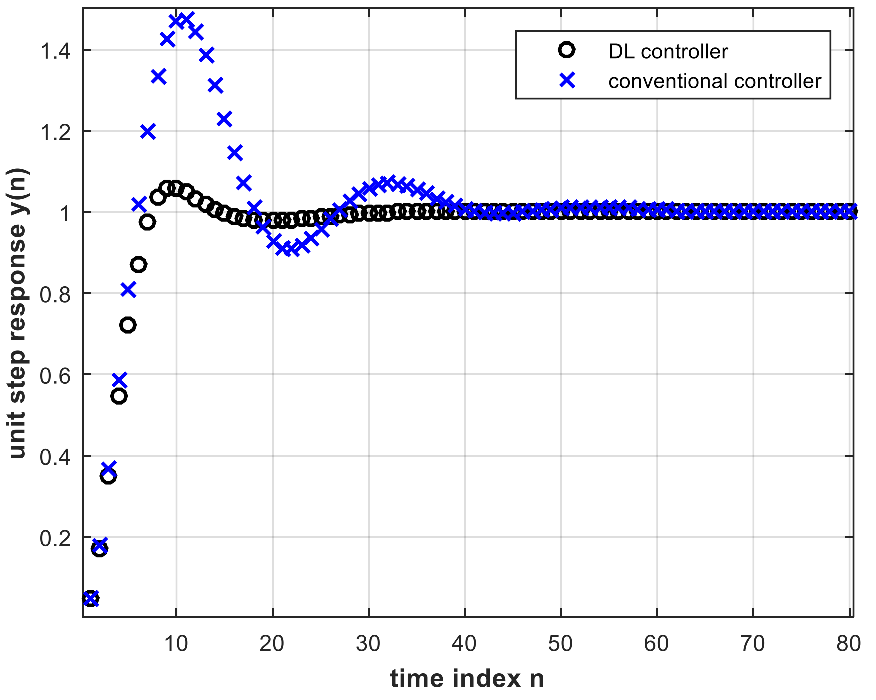

**Figure 11.** A magnified view of Figure 10.

**Table 2.** Unit step response settling times for different controllers under system parameter change.

| **Controller type** | **Settling times (milliseconds) within r % of final value** | | |
|---|---|---|---|
| | r = 2 | r = 0.2 | r = 0.02 |
| Conventional | 40 | 80 | 114 |
| DL with 4 hidden lay-ers | 14 | 30 | 43 |

If the pole locations of the analog plant change further towards instability, the closed loop digital control system unit step response begins to exhibit steady state error while using the same trained DL controller and the same conventional controller. This is shown in Figure 12 and the magnified view of Figure 13 for the analog plant given by:

$$G_p(s)=\frac{6960}{s(s+0.5)} \tag{16}$$

This, together with the zero-order hold, and with T = 0.001sec corresponds to the digital system given by:

$$G(z)=\frac{0.003479\,z+0.003479}{z^2-2\,z+0.9995} \tag{17}$$

Under this parameter change, the analytical expression for the unit step response is:

$$y(n)=1.0901+2.1642(0.9011)^n+3.2810(0.9096)^n\cos(0.0285\pi n+0.9480\pi);\ n\geq 0 \tag{18}$$

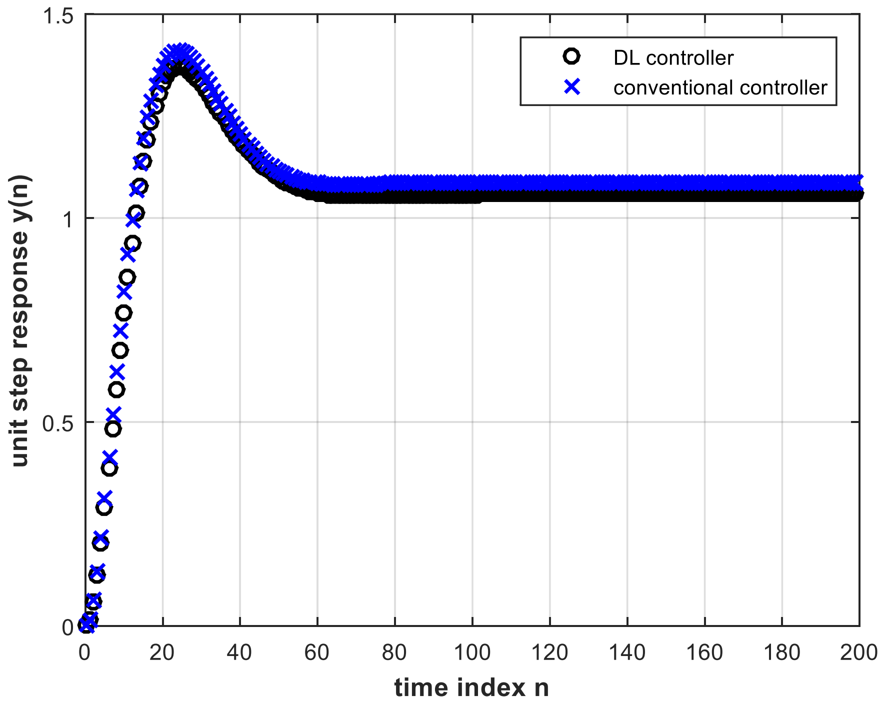

**Figure 12.** Unit step response of the feedback control system with the trained DL controller (four hidden layers and four input nodes) as well as with the conventional controller, both under system parameter change, Equation (17).

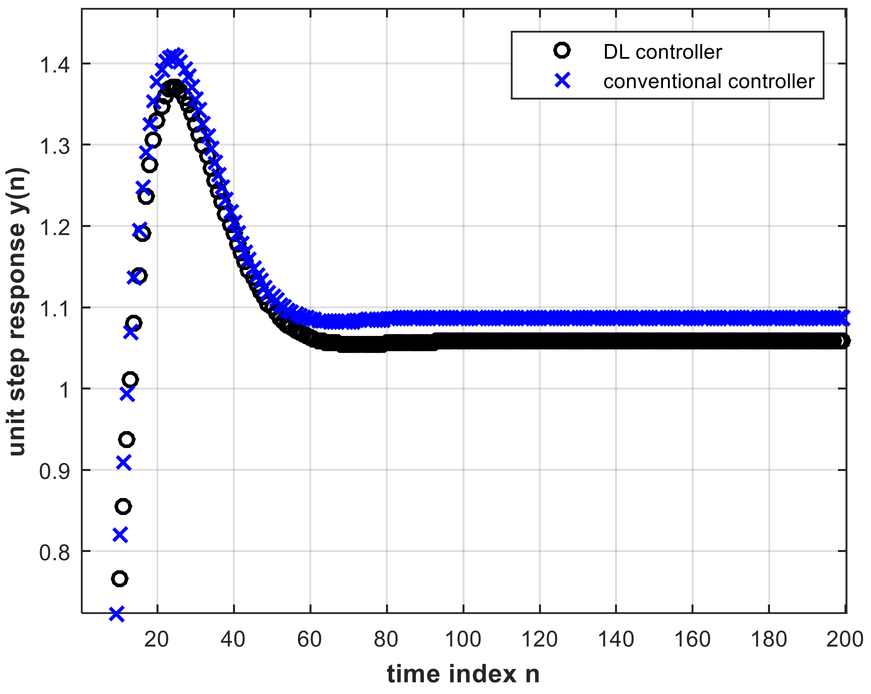

**Figure 13.** A magnified view of Figure 12.

Figure 13 shows a steady state error for the conventional controller that is greater than that for the DL controller which is another advantage of the latter. The DL controller also results in smaller overshoot. By the final value theorem, the steady state error of the digital feedback control system subject to a unit step input, and using the conventional controller, is given by:

$$e_{ss} = \lim_{z \to 1} \left\{ (1 - z^{-1}) \frac{1}{1 + D(z)G(z)} R(z) \right\} \tag{19}$$

where

$$R(z) = \frac{1}{1 - z^{-1}} \tag{20}$$

Therefore,

$$e_{ss} = \lim_{z \to 1} \left\{ \frac{1}{1 + D(z)G(z)} \right\} \tag{21}$$

Substituting in the above by Equations (9) and (17), we find that $e_{ss} = 0.09$. This is in accordance with the curve in Figure 13 that corresponds to the conventional controller and with Equation (18) as n tends to infinity.

## 5. Conclusions

A deep learning controller can efficiently replace a conventional controller in a feedback control system if trained offline with the input-output signals of the conventional controller. It has been shown that, if the DL controller is sufficiently deep, it can outperform the conventional controller in terms of settling time of the step response of the tested second-order system. Also, no re-training is needed under different reference input magnitude or in case of system parameter change. Under system parameter change, the conventional controller needs to be redesigned for comparable performance with the DL controller. Another performance indicator for feedback control systems, namely the steady state error, was also shown to improve using DL controllers.

**Author Contributions:** "Conceptualization, N. A. and Z. H.; methodology, N. A.; software, N. A.; validation, N. A. and Z. H..; formal analysis, Z. H. and N. A.; investigation, N. A. and Z. H..; resources, N. A. and Z. H..; writing—original draft preparation, N. A. and Z. H.; writing—review and editing, Z. H.; visualization, Z. H.; supervision, Z. H.; project administration, Z. H. All authors have read and agreed to the published version of the manuscript."

**Funding:** This research is partially funded by Edith Cowan University via the ASPIRE Program.

**Data Availability Statement:** The MATLAB code is available from the authors on reasonable request.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Louis, F. L.; Ge, S. S. Neural networks in feedback control systems. In *Mechanical Engineers' Handbook*; Kutz, M., Ed.; John Wiley, 2005.
2. Werbos, P. J. Neural network for control and system identification. Proceedings of the 28th IEEE Conf. Decision and Control, 1989, pp. 260-265.
3. Kawato, M. Computational schemes and neural network models for formation and control of multi-joint arm trajectory. In *Neural Networks and Control*; Miller, W. T.; Sutton, R. S.; Werbos, P. J., Eds; MIT Press, 1991.
4. Igelnik, B.; Pao, Y. H. Stochastic choice of basis functions in adaptive function approximation and functional-link net. *IEEE Transactions on Neural Networks* **1995**, *6*, 1320-1329.
5. Rumelhart, D.; Hinton, G.; Williams, R. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533-536.
6. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep learning;* MIT Press, 2016.
7. Kim, P.; *MATLAB Deep Learning;* Apress, 2016.
8. Cheon, K.; Kim, J. H.; Hamadache, M.; Lee, D. On replacing PID controller with deep learning controller for DC motor system. *Journal of Automation and Control Engineering.* **2015**, *3*, 452-456.
9. Aggarwal, C. C., *Neural Networks and Deep Learning.* Springer, 2018.
10. Zaki, A. M.; El-Nagar, A. M.; El-Bardini, M; Soliman, F. A. S. Deep learning controller for nonlinear system based on Lyapunov stability criterion. *Neural Computing and Applications* **2021**, 33, 1515-1531.
11. Lapan, M. *Deep Reinforcement Learning Hands-On*. Packt Publishing Ltd. 2018.
12. Liu, R.; Nageotte, F.; Zanne, P.; de Mathelin, M.; Dresp-Langley, B. Deep reinforcement learning for the control of robotic manipulation: a focused mini-review. *Robotics*, **2021**, 10, 22.
13. Manrique-Escobar, C. A.; Pappalardo, C. M.; Guida, D. A parametric study of a deep reinforcement learning control system applied to the swing-up problem of the cart-pole. *Applied Sciences*, **2020**, 10, 9013.
14. Dorf, R. C.; Bishop, R. H. *Modern Control Systems*, 11th ed.; Pearson Prentice Hall, 2008.
15. Baptista, F. D.; Rodrigues, S.; Morgado-Dias, F. Performance comparison of ANN training algorithms for classification. In Proceedings of the 2013 IEEE 8th International Symposium on Intelligent Signal Processing, 2013, pp. 115-120.
16. Heaton, J. *Introduction to Neural Networks with Java*; Heaton Research Inc., 2008.
17. Vujicic, T.; Matijevic, T.; Ljucovic, J.; Balota, A.; Sevarac, Z. Comparative analysis of methods for determining number of hidden neurons in artificial neural network. In Proceedings of the Central European Conference on Information and Intelligent Systems; Varazdin, Croatia, 2016.
18. Hunter, D.; Yu, H.; Pukish, M. S.; Kolbusz, J.; Wilamowski, B. M. Selection of proper neural network sizes and architectures. *IEEE Transactions on Industrial Informatics* **2012**, *8*, 228-240.
19. Thomas, A. J.; Walters, S. D.; Gheytassi, S. M.; Morgan, R. E.; Petridis, M. On the optimal node ratio between hidden layers: a probabilistic study. *International Journal of Machine Learning and Computing*, **2016**, *6*, 241-247.
20. Sutskever, I.; Martens, J.; Dahl, G.; Hinton, G. On the importance of initialization and momentum in deep learning. In Proceedings of the 30th International Conference on Machine Learning; Atlanta, USA, 2013, pp. 1139-1147.
21. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the 13th International Conference on Artificial Intelligence and Statistics. Sardinia, Italy, 2010, pp. 249-256.
22. Tan, Y.; Dang, X.; Cauwenberghi, A. V. Generalized nonlinear PID controller based on neural networks. In Proceedings of the IEEE Decision and Control Symposium, 1999, pp. 519-524.