In *IEEE International Conference on Mechatronics*, Kashiwa, Japan, 2021

Control-Relevant Neural Networks for Intelligent Motion Feedforward

Leontine Aarnoudse
Eindhoven University of Technology
Eindhoven, The Netherlands
l.i.m.aarnoudse@tue.nl

Wataru Ohnishi
The University of Tokyo
Tokyo, Japan
ohnishi@koseki.t.u-tokyo.ac.jp

Maurice Poot
Eindhoven University of Technology
Eindhoven, The Netherlands
m.m.poot@tue.nl

Paul Tacx
Eindhoven University of Technology
Eindhoven, The Netherlands
p.j.m.m.tacx@tue.nl

Nard Strijbosch
Eindhoven University of Technology
Eindhoven, The Netherlands
n.w.a.strijbosch@tue.nl

Tom Oomen
Eindhoven University of Technology
Eindhoven, The Netherlands
t.a.e.oomen@tue.nl

***Abstract*—Neural networks have large potential for motion feedforward because of their ability to approximate a wide range of functions. The aim of this paper is to develop a systematic framework for application of neural networks to motion feedforward, that leads to an intelligent motion feedforward approach in the sense that it achieves both flexibility for varying references and high performance. Iterative learning control is used to generate training data, and a control-relevant performance function is introduced. Non-causal feedforward is enabled through two network configurations that enable respectively finite and infinite preview. The approach is experimentally validated on an industrial flatbed printer.**

I. Introduction

Neural networks are universal function approximators [1] that have large potential for the feedforward control of motion systems. Neural networks have been applied successfully in a wide range of fields, including speech recognition [2] and image recognition and classification [3]. The increasing popularity of neural networks is enabled by developments in dedicated GPU hardware and cloud-based computing, combined with innovative optimization algorithms [4], [5].

Machine learning and neural networks have also been applied successfully in fields related to control. Examples include reinforcement learning for trajectory following [6] or adaptive control [7], and neural networks for system identification [8], [9]. Neural networks are applied to feedforward control in e.g. [10], where a one-layer network consisting of B-spline basis functions is used to learn a feedforward input for a constant task, and in [11], where a NARX model is used for the inverse modeling of a non-linear thermal mixing process. An early overview of neural networks in control is given in [12].

A key requirement for the training of neural networks is a large amount of data. In [13], for example, 50 million frames of Atari 2600 games are used for deep reinforcement learning. The use of this amount of training data is enabled by progress in computational power and storage, such that nowadays, processing this data is typically not a problem. However, motion feedforward control relies on dynamic systems, where the goal is to generate the input of an unknown dynamic system such that the output follows a certain trajectory. Training neural networks for such systems may require a large amount of training data from a broad range of experimental conditions.

Feedforward signals for motion systems are task-dependent and can be learned iteratively, for example using iterative learning control (ILC) [14], [15]. ILC is capable of obtaining almost perfect feedforward signals for a specific task, but it cannot deal with varying tasks [16]. Basis functions enable task flexibility and high performance [17], [18]. However, these approaches require the selection of suitable basis functions, which is not trivial for general motion systems.

Although recent developments in neural networks and machine learning are promising, currently a systematic framework for application of these techniques to motion feedforward control that integrates typical properties of learning-based feedforward [19] is lacking. The aim of this paper is to develop a framework that takes advantage of the mapping potential of neural networks to achieve both flexibility for varying references and high performance. To achieve this, existing ILC techniques are used to generate training data. In addition, system model knowledge is used to design a control-relevant performance function along similar ideas as in [16], [20]. The non-causal properties of ILC [19] are recovered through choices in network and input design. The contribution of this paper consists of the following cornerstones.

1) The use of neural networks for flexible motion feedforward is proposed, and a framework is introduced that takes advantage of existing user-friendly results.
2) ILC is employed to generate perfect training data.
3) A control-relevant performance function is introduced, to ensure that the neural network is trained towards the goal of feedforward control performance.
4) Non-causal time-delay neural networks are used to enable non-causal feedforward signals with finite preview.
5) The framework is experimentally validated on an industrial flatbed printer.

This paper is organized as follows. In Section II, the problem considered in this paper is formulated. In Section III the

This work is part of the research programme VIDI with project number 15698, which is (partly) financed by NWO.

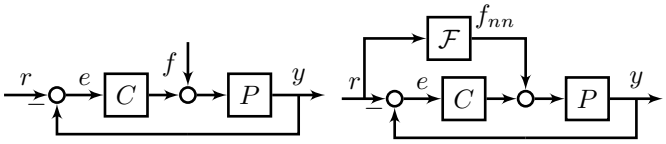

Fig. 1: Standard control setup (left) and control setup with a neural network $\mathcal{F}(r)$ that maps r tof_{nn} (right).

proposed framework for neural network motion feedforward is presented. The framework is experimentally validated in Section IV. Conclusions are given in Section V.

II. Problem formulation

In this section, feedforward control for motion systems is introduced. Then, artificial neural networks are introduced and lastly, neural networks for motion feedforward are proposed.

A. Feedforward control for motion systems

The goal of feedforward control is to compensate for known disturbances using an accurate feedforward signal. Consider the SISO control system shown in Fig. 1 (left). The error signal $e \in \mathbb{R}$ is given by

$$e = Sr - SPf \tag{1}$$

with reference $r \in \mathbb{R}$, feedforward signal $f \in \mathbb{R}$ and system sensitivity $S = \frac{1}{1+PC}$ with plant P and controller C. Since the reference r is a known disturbance, it can be compensated using f. The feedforward signal for which $e = 0$ is given by

$$f = (SP)^{-1}Sr. \tag{2}$$

Since SP is typically not know exactly, the feedforward signal that results in a zero error cannot be determined directly. For repeating tasks, f can be learned iteratively using iterative learning control (ILC), see e.g. [14]. Alternatively, for systems that require task flexibility, f can be approximated using basis functions that are based on a system model that typically includes acceleration and snap feedforward, as well as compensation for Coulomb and viscous friction [21]. This results in a basis function feedforward signal, given by

$$f_{\text{bf}} = \psi(r)^\top \theta \tag{3}$$

with $\psi(r) = \begin{bmatrix}\psi_1(r) & \psi_2(r) & \dots & \psi_N(r)\end{bmatrix}^\top$ a set of basis functions and $\theta = \begin{bmatrix}\theta_1 & \theta_2 & \dots & \theta_N\end{bmatrix}^\top$ a parameter vector. The parameters can be tuned manually during experiments or automatically using ILC with basis functions (BF-ILC) [18].

B. Artificial neural networks

Artificial neural networks are universal function approximators that are capable of finding complicated, possibly nonlinear mappings between signals. A neural network consists of neurons with multiple inputs and a single output. The neurons are collected in layers, such that the outputs of the neurons in one layer form the inputs for the neurons in the next layer.

In this paper, two types of neural networks are considered. The first is the feedforward neural network (FNN), an example of which is shown in Fig. 2. In an FNN, the information moves

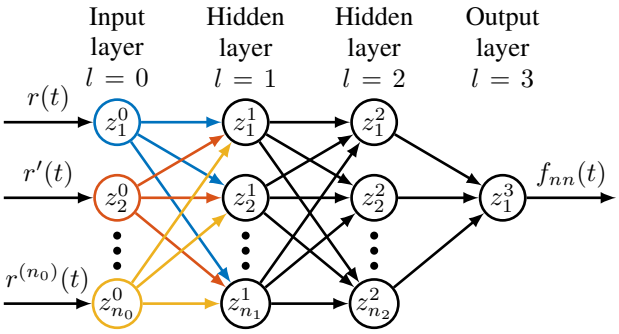

Fig. 2: FNN with two fully connected hidden layers.

in one direction from the input nodes, through the hidden layers that consist of neurons, to the output nodes. The output of a neuron z_k^l, the k^{th} neuron in layer l of a neural network, is given by

$$z_k^l = \sigma\left(\textstyle\sum_{j=1}^{n_{l-1}} w_{kj}^l z_j^{l-1} + b_k^l\right) \tag{4}$$

with $w_{kj}^l \in \mathbb{R}$ the weights, $b_k^l \in \mathbb{R}$ the biases, n_{l-1} the number of neurons in the previous layer and σ an activation function, which is a possibly nonlinear mapping. The hyperparameters for an FNN include the number of neurons per layer, the number of hidden layers, the connections between the layers and the activation functions. FNNs can be interpreted as nonlinear autoregressive exogenous (NARX) models [9].

The second type of neural network considered in this paper is the time-delay neural network (TDNN), which is typically implemented as an FNN. However, unlike in a standard FNN, the neurons in a TDNN obtain inputs from a window of outputs from the previous layer, see also Section III-C.

C. Neural networks for motion feedforward

The aim of motion feedforward control is to minimize the error e in (1) in the presence of a known but varying exogenous disturbance r. This paper aims to exploit the ability of neural networks to find mappings without system preknowledge to achieve both high performance and flexibility for varying references in motion feedforward. Specifically, the neural network aims to achieve flexibility for varying tasks by finding a mapping $\mathcal{F}(r)$ between the reference signal r and a feedforward signal f_{nn}, such that

$$f_{\text{nn}} = \mathcal{F}(r), \tag{5}$$

and the error

$$e_{\text{nn}} = Sr - SPf_{\text{nn}} \tag{6}$$

is minimized. The corresponding control setup is illustrated in Fig. 1 (right). To achieve high performance, a system-specific control-relevant performance function is designed that is aimed at minimizing the system error, and the neural nets and inputs are designed to enable non-causal feedforward actions.

III. Neural networks for motion feedforward

In this section, the framework for neural networks for motion feedforward is introduced. The aim of using neural networks for motion feedforward is to find a mapping $\mathcal{F}(r)$ that

minimizes e_{nn} in (6). In this section the focus is on finding the parameters of $\mathcal{F}$, i.e., the training of the network independent from its design and configuration. First, the data acquisition and general approach are explained. Then, a control-relevant performance function is introduced and lastly, the possibility of including non-causal feedforward in f_{nn} is discussed.

A. Data acquisition and general approach

Large amounts of data are required to train a neural network. To find the mapping $\mathcal{F}$ in (5), this data consists of reference signals r and corresponding feedforward signals f that minimize e in (1).

To generate the required training data, iterative learning control (ILC) is used. A set of N references r_i, $i = 1, 2, ..., N$ is defined that well covers the set of possible references in terms of positions, directions, velocities, accelerations, and jerk and snap values. Standard ILC [14] is used to find the feedforward signals $f_{\text{train},i}$ corresponding to these references.

To train the neural network the ILC data is randomly separated into training, validation and test sets. The validation set is used during training to check if the network is generalizing well. The final performance of the network is evaluated using the test set, which in case of motion feedforward can involve testing on the real system. The training of a neural network involves the minimization of a performance function $\mathcal{J}(f_{\text{nn}})$, see Section III-B, typically using a backpropagation algorithm [22, Chapter 16.5]. The specific algorithm used for neural network motion feedforward depends on the performance function and network, see Section IV. The complete procedure for neural network motion feedforward is described in Algorithm 1.

Algorithm 1 Neural Network Motion Feedforward

1: Design a representative set of references.
2: Apply standard ILC and find the optimal feedforward input for each of the references.
3: Separate the data in training, validation and test sets.
4: Train the network: minimize $\mathcal{J}(f_{\text{nn}})$ (Section III-B) and evaluate the performance on the validation set.
5: Evaluate the performance using the test set.
6: Use the neural network to generate feedforward signals for arbitrary references.

B. Control-relevant performance function

The main requirement for neural networks for motion feedforward is the ability to generate a feedforward signal f_{nn} that achieves a small system error e_{nn} in (6). In view of this requirement, a new criterion is formulated for control-oriented training of neural networks. Since this new criterion can lead to large changes compared to existing approaches, a regularization term is introduced to avoid undesirable solutions.

It is assumed that the training feedforward signals that are obtained using standard ILC result in a zero error:

Assumption 1 (Perfect training data)**.** *For each reference r_i, the corresponding feedforward signal $f_{\text{train},i}$ results in zero error, i.e.,*

$$e_i = Sr_i - SPf_{\text{train},i} = 0. \tag{7}$$

This can be achieved by standard ILC provided that there are no trial-varying disturbances present.

A performance function that reflects the goal of minimizing e_{nn} by taking the system dynamics into account is proposed in [20] and involves filtering the fitting error

$$e_{\text{fit}} = f_{\text{train}} - f_{\text{nn}} \tag{8}$$

through the process sensitivity SP of the system. A similar performance function is proposed for neural network motion feedforward. For motion systems with integrator action in the controller, SP is typically small at low frequencies. As a result, the term SPe_{fit} does not entirely capture constant offsets in e_{fit}, which could lead to undesired effects in case of nonlinear effects in the motion system. Regularization is added to the control-relevant performance function to prevent such offsets. Under Assumption 1, the following theorem holds.

Theorem 2. *The feedforward signal f_{nn} that results in the smallest error e_{nn} in (6) in terms of the squared 2-norm is the minimizer of the performance function*

$$\mathcal{J}(f_{\text{nn}}) = \|SP(f_{\text{train}} - f_{\text{nn}})\|_2^2, \tag{9}$$

with the two-norm defined as $\|x\|_2 = \sqrt{x^\top x}$. Adding a regularization term prevents a constant offset in $f_{\text{train}} - f_{\text{nn}}$, and results in the control-relevant performance function

$$\mathcal{J}_{CR}(f_{\text{nn}}) = w_1\|SP(f_{\text{train}} - f_{\text{nn}})\|_2^2 + w_2\|f_{\text{train}} - f_{\text{nn}}\|_2^2. \tag{10}$$

with scalar weights w_1 and w_2.

Proof. Under Assumption 1, it holds that

$$f_{\text{train},i} = (SP)^{-1}Sr_i. \tag{11}$$

Using (11), (6) can be rewritten to

$$e_{\text{nn}}(f_{\text{nn}}) = SPf_{\text{train}} - SPf_{\text{nn}} = SP(f_{\text{train}} - f_{\text{nn}}), \tag{12}$$

Taking $\|e_{\text{nn}}\|_2^2$ results in the performance function

$$\mathcal{J}(f_{\text{nn}}) = \|SP(f_{\text{train}} - f_{\text{nn}})\|_2^2. \tag{13}$$

The regularization term prevents an offset in e_{fit} in (8). □

In practice SP is typically unknown and a model is used instead, in which case the regularization term also provides robustness against model uncertainty. Note that the standard mean-squared error (MSE) performance function

$$\mathcal{J}_{\text{MSE}}(f_{\text{nn}}) = \|f_{\text{train}} - f_{\text{nn}}\|_2^2. \tag{14}$$

that is common in neural network training is ill-suited for motion feedforward purposes, as it only minimizes the regularization term without considering the system performance.

C. Non-causal neural networks

The inverse of the process sensitivity $(SP)^{-1}$, which is part of the mapping from r to f in (2), is typically non-causal for motion systems because the plant P is strictly proper or non-minimum phase. Two different neural network configurations can recover non-causal feedforward with respectively finite and infinite preview.

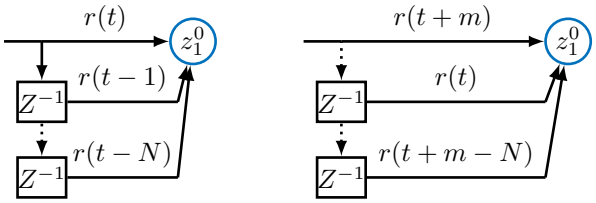

Fig. 3: Input node for causal (left) and non-causal (right) time-delay neural networks with N input delays.

To obtain a feedforward signal with finite preview, a time-delay neural network with shifted inputs is proposed. A standard TDNN [23] can map a finite time sequence of the reference $\{r(t), r(t-1), ..., r(t-N)\}$ to a single feedforward input $f_{\text{nn}}(t)$. This causal operation uses only previous reference samples. To obtain non-causal inputs with finite preview, a shifted reference signal is used as input. Take as input $r_{\text{shift}}(t) = r(t+m)$, such that $\{r_{\text{shift}}(t), r_{\text{shift}}(t-1), ..., r_{\text{shift}}(t-N)\} = \{r(t+m), r(t+m-1), ..., r(t+m-N)\}$. The TDNN can find a mapping between $r_{\text{shift}}(t)$, which includes finite preview, and $f_{\text{nn}}(t)$. Input nodes for causal and non-causal TDNNs are illustrated in Fig. 3.

In the broader picture regarding preview, one can also include infinite preview, see e.g. [19] on inversion for feedforward. To achieve non-causal feedforward with infinite preview, a recurrent neural network (RNN) with a bi-directional long short-term memory (BLSTM) layer [24] can be used. LSTM layers contain a hidden 'state' with information on prior inputs and outputs, and are suitable for series inputs with no predetermined size. A BLSTM layer contains two sets of neurons, that receive respectively forward and time-reversed data. A BLSTM layer has access to all past and future data, and is therefore capable of generating feedforward signals with infinite preview. In view of space limitations, only finite preview is shown here. Design details and experimental results with infinite time preview will be published elsewhere.

D. Recovering ILC with polynomial basis functions

ILC with polynomial basis functions is often used to design feedforward signals for motion systems with varying references. Consider the parameterized feedforward signal in (3). A typical choice for polynomial basis functions in ILC is to take derivatives of the reference signal, i.e., $\psi(r) = \begin{bmatrix} r & r' & r'' & r^{(3)} & r^{(4)} \end{bmatrix}^\top$. These functions have a physical interpretation, e.g., r'' and $r^{(4)}$ provide respectively mass and snap feedforward. Basis function ILC can be recovered by a neural network that uses only the position reference as input.

To illustrate this, consider an FNN that uses five reference samples, $r(t)$ up to $r(t+4)$, to determine $f_{\text{nn}}(t)$. With these reference samples the derivative of r can be estimated numerically up to $r^{(4)}$ using forward difference methods, where the m^{th} derivative of $r(t)$ is approximated by a weighted sum of the reference samples, given by

$$\hat{r}^{(m)}(t) = \sum_{j=0}^{m} (-1)^{m-j} \frac{m!}{j!(m-j)!} r(t+j). \qquad (15)$$

The output for neuron z_k^l in a fully connected layer l of a neural network is given in (4). Take the inputs as $z_k^0 = r(t+k-1)$, take the bias for layer $l=1$ as $b_k^1 = 0\,\forall k$ and choose the activation function σ to be a simple linear mapping. Then,

$$z_k^1 = \sum_{j=1}^{n_0} w_{kj}^1 z_j^0 = \sum_{j=1}^{n_0} w_{kj}^1 r(t+j-1). \qquad (16)$$

For certain combinations of weights, (16) can recover each of the derivative estimates $\hat{r}^{(m)}(t)$ in (15). Thus, by using five consecutive reference samples as input for each time step, and an FNN with one fully connected layer with five neurons, it is possible to recover $\psi(r)$.

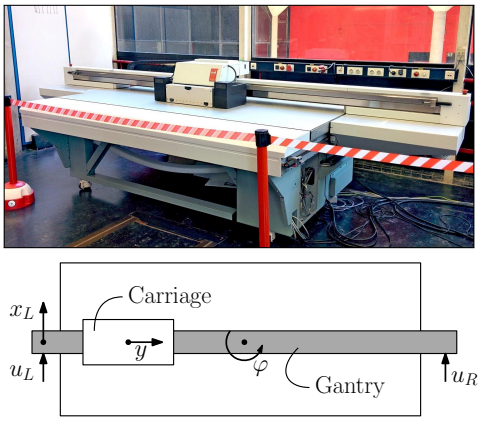

Fig. 4: Photo (top) and schematic overview (bottom) of the Arizona flatbed printer.

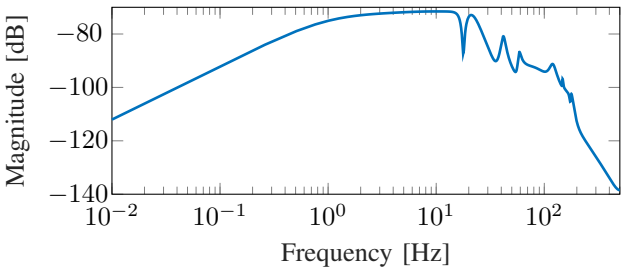

Fig. 5: Process sensitivity SP of the y-axis of the printer.

IV. Experimental results

In this section the proposed approach is experimentally validated. In Section IV-A, the setup is introduced and the approach is explained. Then, four aspects of neural network motion feedforward are experimentally validated: the benefits of a control-relevant performance function; finite preview using a non-causal TDNN; comparison of neural network motion feedforward to basis function ILC; and the possibility to recover polynomial basis functions using an FNN.

A. Experimental setup and approach

The proposed approach is validated using an industrial flatbed printer, shown in Fig. 4. A SISO situation is considered, in which input F_y is the force on the carriage and output y is the position of the carriage on the gantry. The gantry position and orientation are kept constant. A Bode magnitude plot of the process sensitivity of the y-axis is shown in Fig. 5.

The approach of Algorithm 1 is applied. Ten different references are designed and frequency domain ILC is used to generate a feedforward signal for each of these references.

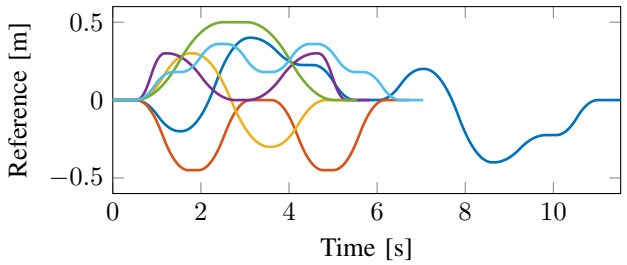

Fig. 6: Reference signals used for testing (—) and some of those used for training (—,—,—,—,—).

The resulting combination of references and corresponding feedforward signals is used to train the neural networks, which are then tested on an eleventh reference that is not part of the training set. Five of the training references as well as the testing reference are shown in Fig. 6.

All networks, except for the FNN that recovers BF-ILC, take as input the reference with its first four derivatives and the sign of the velocity. The neural networks are generated using MATLAB's Deep Learning Toolbox, function names are given between brackets. The training algorithm that is used depends on the performance function. In case of $\mathcal{J}_{\text{CR}}$, resilient backpropagation (`trainrp`) is used, while for $\mathcal{J}_{\text{MSE}}$ the more efficient Levenberg-Marquardt backpropagation (`trainlm`) is available. Both MATLAB functions randomly separate the data in training and validation sets.

Remark. For all networks, the given hyperparameters, i.e., the number of neurons, hidden layers and the type of transfer functions, are empirically tuned and could be optimized further.

B. Control-relevant performance function

A feedforward neural network (`feedforwardnet`), consisting of two hidden layers with ten neurons each using a linear transfer function (`purelin`) as shown in Fig. 2, is trained using two different performance functions: the control-relevant performance function $\mathcal{J}_{\text{CR}}$ (10) with $w_1 = 10^2$ and $w_2 = 10^{-2}$, and a standard mean-squared error performance function $\mathcal{J}_{\text{MSE}}$ (14).

In Table I, the resulting 2-norm $\|e_{\text{nn}}\|_2 = \sqrt{e_{\text{nn}}^\top e_{\text{nn}}}$ and ∞-norm $\|e_{\text{nn}}\|_\infty = \max_i |e_{\text{nn}}(i)|$ are shown. Note that although the networks were trained using a 2-norm based performance function, the ∞-norm is also a relevant indication of performance for motion systems. When using $\mathcal{J}_{\text{CR}}$ instead of $\mathcal{J}_{\text{MSE}}$, the error 2-norm, the model-based approximation of which is minimized in $\mathcal{J}_{\text{CR}}$, is reduced by 1.5%, while the ∞-norm is reduced by 3.5%, showing the feasibility of using $\mathcal{J}_{\text{CR}}$.

C. Non-causal TDNNs

To analyze the advantages of non-causal feedforward, the performance of causal and non-causal TDNNs is compared. Due to implementation limitations, an MSE performance function (14) is used. The following hyperparameters are used.

1) TDNN (`timedelaynet`): ten input delays, one hidden layer with fifteen neurons and a tan-sigmoid transfer function (`tansig`), and an output layer with a linear transfer function (`purelin`).

TABLE I: Overview of error 2-norm and maximum absolute value for different performance functions.

Approach	$\mathcal{J}$	$\|e_{\text{nn}}\|_2$	$\|e_{\text{nn}}\|_\infty$
FNN	$\mathcal{J}_{\text{MSE}}$	$1.840 \cdot 10^{-3}$	$9.04 \cdot 10^{-5}$
FNN	$\mathcal{J}_{\text{CR}}$	$1.814 \cdot 10^{-3}$	$8.74 \cdot 10^{-5}$

TABLE II: Overview of error 2-norm and maximum absolute value for different approaches.

Approach	$\|e_{\text{nn}}\|_2$	$\|e_{\text{nn}}\|_\infty$
BF-ILC	$1.836 \cdot 10^{-3}$	$9.40 \cdot 10^{-5}$
FNN	$1.840 \cdot 10^{-3}$	$9.04 \cdot 10^{-5}$
TDNN	$1.764 \cdot 10^{-3}$	$8.36 \cdot 10^{-5}$
Non-causal TDNN	$1.632 \cdot 10^{-3}$	$6.17 \cdot 10^{-5}$

2) Non-causal TDNN (`timedelaynet`): twenty input delays, one hidden layer with fifteen neurons and a tan-sigmoid transfer function (`tansig`), and an output layer with a linear transfer function (`purelin`). The input is shifted over ten samples such that $r_{\text{shift}}(t) = r(t+10)$.

The causal and non-causal TDNNs are identical except for the ten future reference samples available to the non-causal TDNN. The non-causal TDNN reduces the error 2-norm by 7.5% and the error ∞-norm by 26% compared to the causal TDNN, as shown in Table II.

D. Comparison between approaches

The FNN and TDNNs of Section IV-B and IV-C are compared to polynomial basis functions that are typically used in ILC, given by $\psi(r) = \begin{bmatrix} r & r' & \text{sign}(r') & r'' & r^{(3)} & r^{(4)} \end{bmatrix}^\top$. An MSE performance function (14) is used.

In Table II the obtained error 2-norm and ∞-norm are shown for the different approaches. A simple FNN is shown to be able to match the performance of BF-ILC. A standard TDNN improves the 2-norm by 4% and the ∞-norm by 11%. The best performance is achieved by the non-causal TDNN, which improves the 2-norm by 11% and the ∞-norm by 34%.

The feedforward signals generated by BF-ILC and the non-causal TDNN, and the corresponding error signals, are shown in Fig. 7. The input signal generated by the TDNN is more complicated, and the resulting error signal is smaller overall and has lower peak values. It is concluded that neural network feedforward can achieve an increased performance compared to BF-ILC when using references outside of the training set.

E. Recovering basis functions

As shown in Section III-D, it is possible to recover a set of polynomial basis functions $\psi(r) = \begin{bmatrix} r & r' & r'' & r^{(3)} & r^{(4)} \end{bmatrix}^\top$ as used in BF-ILC using an FNN with one fully connected hidden layer with five neurons and a linear transfer function. The input of the FNN for each time instant t consists of $r(t)$ up to $r(t+4)$. The approaches generate similar feedforward signals, and Table III shows that the achieved error norms for these two approaches are similar as well.

V. Conclusions

A systematic approach to using neural networks for motion feedforward is introduced, that is capable of achieving both

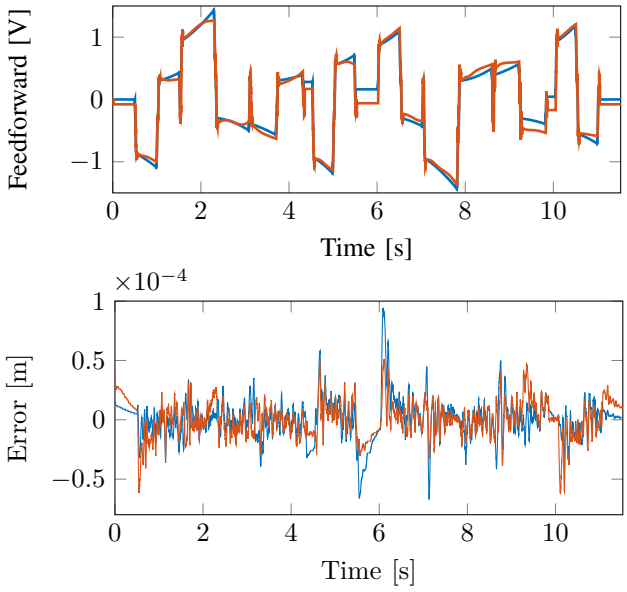

Fig. 7: Feedforward (a) and error signals (b) for the testing reference using BF-ILC (—) and a non-causal TDNN (—).

TABLE III: Overview of error 2-norm and maximum absolute value when recovering BF-ILC.

Approach	$\|e_{nn}\|$	$\|e_{nn}\|_\infty$
BF-ILC (derivatives of r)	$2.255 \cdot 10^{-3}$	$12.81 \cdot 10^{-5}$
FNN to recover BF-ILC	$2.260 \cdot 10^{-3}$	$12.60 \cdot 10^{-5}$

flexibility and high performance. Iterative learning control is used to generate training data. A control-relevant performance function is proposed that takes into account the system response when training the neural network, and non-causal feedforward with finite preview is enabled through a time-delay neural network with shifted inputs. Experimental validation on an industrial flatbed printer shows the benefits of using a control-relevant performance function and non-causal feedforward with finite preview. Neural network motion feedforward in general is shown to be capable of achieving improved performance for flexible motion feedforward compared to existing basis function ILC techniques. In future work, the performance of RNNs with a BLSTM layer to enable infinite preview should also be experimentally validated.

Acknowledgment

The authors gratefully acknowledge the contributions to this paper through a new challenge-based learning project by Lennard Ceelen, Spyros Chatzizacharias, Mathyn van Dael, Yves Elmensdorp, Gijs Herings, Jilles van Hulst, Laurens Kools, Martijn Kortenhoeven, Mike Mostard, Bart Reijnen, Pim Scheers, Matthijs Schotman, Stan Verbeek, Joey Verdonschot, Peter Verheijen, and Jelle de Vries.

The authors also wish to thank Koen Tiels and Sjirk Koekebakker for their contributions. Marcel Heertjes is gratefully acknowledged for inspiring discussions on the topic.

References

[1] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Process. Mag.*, vol. 82, pp. 82–97, 2012.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and A. C. Berg, "ImageNet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vis.*, vol. 115, pp. 211–252, 2015.

[4] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd Int. Conf. Learn. Represent.*, 2015.

[5] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, 2018.

[6] A. Coates, P. Abbeel, and A. Y. Ng, "Learning for Control from Multiple Demonstrations," in *25th Int. Conf. Mach. Learn.*, Helsinki, Finland, 2008, pp. 144–151.

[7] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *IEEE Control Syst. Mag.*, vol. 32, no. 6, pp. 76–105, 2012.

[8] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 4–27, 1990.

[9] L. Ljung, C. Andersson, K. Tiels, and T. B. Schön, "Deep learning and system identification," in *21st IFAC World Congr.*, 2020.

[10] G. Otten, T. J. A. De Vries, J. Van Amerongen, A. M. Rankers, and E. W. Gaal, "Linear motor motion control using a learning feedforward controller," *IEEE/ASME Trans. Mechatronics*, vol. 2, no. 3, pp. 179–187, 1997.

[11] O. Sørensen, "Additive feedforward control with neural networks," in *14th Trienn. IFAC World Congr.*, Beijing, China, 1999, pp. 1378–1383.

[12] K. Hunt, D. Sbarbaro, R. Zbikowski, and P. Gawthrop, "Neural networks for control systems - A survey," *Automatica*, vol. 28, no. 6, pp. 1083–1112, 1992.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[14] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *IEEE Control Syst.*, vol. 26, no. 3, pp. 96–114, 2006.

[15] T. Oomen, "Learning for advanced motion control," in *IEEE Int. Work. Adv. Motion Control*, Agder, Norway, 2020.

[16] T. Oomen and C. R. Rojas, "Sparse iterative learning control with application to a wafer stage: Achieving performance, resource efficiency, and task flexibility," *Mechatronics*, vol. 47, pp. 134–147, 2017.

[17] J. Bolder, T. Oomen, S. Koekebakker, and M. Steinbuch, "Using iterative learning control with basis functions to compensate medium deformation in a wide-format inkjet printer," *Mechatronics*, vol. 24, no. 8, pp. 944–953, 2014.

[18] L. Blanken, F. Boeren, D. Bruijnen, and T. Oomen, "Batch-to-batch rational feedforward control: From iterative learning to identification approaches, with application to a wafer stage," *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 2, pp. 826–837, 2017.

[19] J. van Zundert and T. Oomen, "On inversion-based approaches for feedforward and ILC," *Mechatronics*, vol. 50, no. November 2016, pp. 282–291, 2018.

[20] F. Boeren, A. Bareja, T. Kok, and T. Oomen, "Frequency-Domain ILC Approach for Repeating and Varying Tasks: With Application to Semiconductor Bonding Equipment," *IEEE/ASME Trans. Mechatronics*, vol. 21, no. 6, pp. 2716–2727, 2016.

[21] T. Oomen, "Control for precision mechatronics," in *Encycl. Syst. Control*, 2nd ed., J. Baillieul and T. Samad, Eds. London: Springer, 2019.

[22] K. P. Murphy, *Machine Learning: a probabilistic perspective*. Cambridge, Massachusetts: The MIT Press, 2012.

[23] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme Recognition Using Time-Delay Neural Networks," *IEEE Trans. Acoust.*, vol. 37, no. 3, pp. 328–339, 1989.

[24] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, pp. 602–610, 2005.