Finite-Set Direct Torque Control via Edge-Computing-Assisted Safe Reinforcement Learning for a Permanent-Magnet Synchronous Motor

Maximilian Schenke, Barnabas Haucke-Korber, *Member, IEEE*, and Oliver Wallscheid, *Senior Member, IEEE*

***Abstract*—Advances in the field of reinforcement learning (RL)-based drive control allow formulation of holistic optimization goals for the data-driven training phase. The resulting controllers feature efficient drive operation without the necessity of an a priori known plant model but, so far, conduction of the corresponding training phase in real-world drive systems has been applied only sparsely due to safety concerns. This contribution targets the challenging problem of self-learning torque control for a permanent-magnet synchronous motor assuming a finite control set, i.e., the direct selection of switching actions instead of a modulator-based setup. In order to allow a secure and effective online training with real-world drive systems, the RL controller is monitored by a safeguarding algorithm that prevents application of unsafe switching actions, e.g., such that result in overcurrent. The accruing amount of measurement data is handled with the use of an edge-computing pipeline to outsource the RL training from the embedded control hardware. The inference of the utilized artificial neural network in hard real time is realized with the use of a reconfigurable field-programmable gate array architecture. The resulting RL-based algorithm is able to learn a torque control policy in just 10 min, which has been validated during comprehensive real-world experiments.**

***Index Terms*—Data-driven optimal control, deep reinforcement learning (RL), direct torque control (DTC), edge computing, electric drive, field-programmable gate array (FPGA), Internet of Things, neural network, safe learning, synchronous motors, system identification.**

I. Introduction

DUE to their extensive range of applications and high power density, permanent-magnet synchronous motor (PMSM) drives have earned popularity, not only for industrial utilization but also in electric mobility [1]. The wide distribution of these drive systems motivated a sophisticated but also complicated control theory that has been an important research topic ever since. Ranging from linear, field-oriented control schemes [2] over direct torque control (DTC) methods [3] to model predictive control (MPC) approaches [4], model-based design approaches have been established in drive control for more than 40 years. In the recent past, however, data-driven controller design procedures like reinforcement learning (RL) are increasingly prominent as contemporary hardware allows handling of large scale data [5].

In contrast to the established model-based optimal control techniques, which require significant design effort by human experts, RL-based solutions automatically learn how to control an arbitrary drive system without any human intervention. Especially in the face of labor shortage, such data-driven automation will contribute to ensure high productivity in research and development processes.

A. State of the Art

Recent investigations on RL-based motor control schemes have broadened the range of available controller design procedures and allow a different perspective on the field of optimal drive control.

1) The optimal drive control policy is learned during experiments. Consequently, an a priori drive model (i.e., specific system knowledge) is not needed.
2) Iron losses, magnetic (cross-)saturation and other parasitic effects are indirectly considered within the control scheme as they are affecting the measurements, and therefore, the data being used for RL-based control.
3) Multiple objectives can be considered within the same optimal controller on an infinite time horizon.

For the scenario of PMSM current control, the mentioned points have been successfully validated in publications [6], [7], [8]. However, the considered current control task is only an intermediate step when control of mechanical quantities (torque, speed, or position) is targeted. Consequently, an operation strategy for the motor currents (i.e., controlling the motor current such that the targeted mechanical behavior results) can be incorporated into the design of an RL torque controller as well. This approach has been theoretically discussed in [9] under the label

Manuscript received 7 February 2023; revised 22 June 2023; accepted 5 August 2023. Date of publication 9 August 2023; date of current version 22 September 2023. This work was supported in part by the German Research Foundation (DFG) under Grant 459524199, and in part by the EU Horizon 2020 project VEDLIoT under Grant 957197. Recommended for publication by Associate Editor R. Kennel. *(Corresponding author: Maximilian Schenke.)*

The authors are with the Department of Power Electronics and Electrical Drives, Paderborn University, 33098 Paderborn, Germany (e-mail: schenke@lea.uni-paderborn.de; haucke-korber@lea.uni-paderborn.de; wallscheid@lea.uni-paderborn.de).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TPEL.2023.3303651.

Digital Object Identifier 10.1109/TPEL.2023.3303651

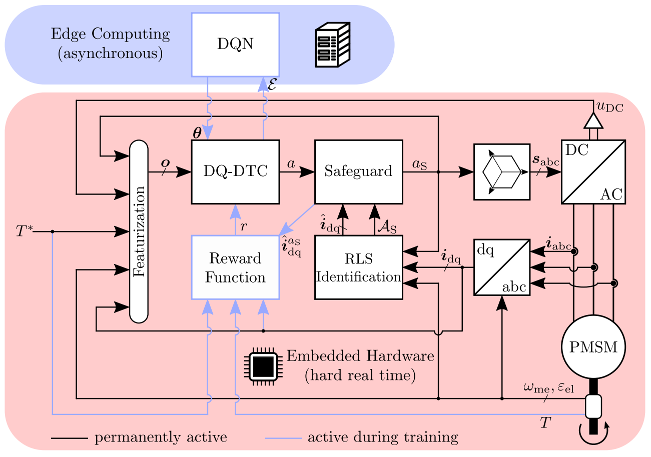

Edge Computing (asynchronous)
DQN
$\mathcal{E}$
$\boldsymbol{\theta}$
u_{DC}
Featurization
$\boldsymbol{o}$
DQ-DTC
a
Safeguard
a_{S}
$\boldsymbol{s}_{\mathrm{abc}}$
DC
AC
T^*
r
$\hat{\boldsymbol{i}}_{\mathrm{dq}}^{a_{\mathrm{S}}}$
$\hat{\boldsymbol{i}}_{\mathrm{dq}}$
$\mathcal{A}_{\mathrm{S}}$
Reward Function
RLS Identification
$\boldsymbol{i}_{\mathrm{dq}}$
dq
abc
$\boldsymbol{i}_{\mathrm{abc}}$
PMSM
Embedded Hardware (hard real time)
$\omega_{\mathrm{me}}, \varepsilon_{\mathrm{el}}$
T
permanently active
active during training

Fig. 1. Schematic of the safeguarded DQ-DTC structure.

deep Q DTC (DQ-DTC). A practical validation of the described concept as well as further methodological improvements are to be presented within this article.

B. Contribution

Despite the aforementioned benefits of RL-based control, there are still several issues to be investigated to establish safe and fast applicability in the real world. Therefore, this article proposes the following contributions in order to improve the usability of the DQ-DTC.

1) A safeguarding layer that ensures adherence to all operation limits (i.e., current and voltage constraints) at runtime in order to avoid corresponding plant system downtimes. This is particularly important as RL algorithms require (random) exploratory actions during the training process, which could lead to overloading the drive systems.
2) An edge-computing, online-learning pipeline that enables training of the DQ-DTC using test bench measurements.
3) A resource-efficient and online-reconfigurable field-programmable gate array (FPGA) implementation of the necessary artificial neural network (ANN), which enables real-time capable policy inference on the control hardware.
4) A fully automated, fast RL framework delivering an expedient, data-driven torque control policy in just a single digit number of minutes without the need of any a priori plant model knowledge [10].[1]

Most importantly, comprehensive test bench experiments are performed in order to prove the feasibility of the DQ-DTC in practice. Further, all scripts and programs created within the scope of this investigation are openly published [11].[2] A schematic of the proposed control scheme is depicted in Fig. 1 with detailed explanations for each individual component in the following sections.

[1]The training process is presented in this video: https://www.youtube.com/watch?v=hQ49Mc6LV78

[2]The implementation details can be found in this open-source repository: http://www.github.com/max-schenke/EdgeRL

II. Drive System

The drive system under investigation features the utilization of a three-phase two-level voltage source inverter and a PMSM. The combination of these components is a standard setup that can be found in many industrial and automotive applications [1]. Both are to be presented shortly in the following.

A. Permanent-Magnet Synchronous Motor (PMSM)

The PMSM is a standard component of modern drive systems. Especially highly utilized PMSMs feature a high torque density and are, therefore, prevalent in applications where lightweight and space-saving motors are required [12]. Characterization of the PMSM can be simplified by utilizing well-known coordinate transformations

$$\underbrace{\begin{bmatrix} x_{\mathrm{d}} \\ x_{\mathrm{q}} \end{bmatrix}}_{\boldsymbol{x}_{\mathrm{dq}}} = \begin{bmatrix} \cos(\varepsilon_{\mathrm{el}}) & \sin(\varepsilon_{\mathrm{el}}) \\ -\sin(\varepsilon_{\mathrm{el}}) & \cos(\varepsilon_{\mathrm{el}}) \end{bmatrix} \underbrace{\begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \end{bmatrix} \overbrace{\begin{bmatrix} x_{\mathrm{a}} \\ x_{\mathrm{b}} \\ x_{\mathrm{c}} \end{bmatrix}}^{\boldsymbol{x}_{\mathrm{abc}}}}_{\boldsymbol{x}_{\alpha\beta} = \begin{bmatrix} x_{\alpha} & x_{\beta} \end{bmatrix}^{\top}}. \quad (1)$$

Herein, the physical three-phase quantities $\boldsymbol{x}_{\mathrm{abc}}$ are reinterpreted as 2-D, orthogonal coordinates that can either be viewed from the stator-fixed αβ reference frame, or from the rotor-fixed dq reference frame, which is defined by rotating the stator-fixed quantities $\boldsymbol{x}_{\alpha\beta}$ by the electrical rotor angle $\varepsilon_{\mathrm{el}}$. Rotor-fixed coordinates allow a compact definition of the PMSM's dynamic behavior[3]

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\psi}_{\mathrm{dq}}(t) = \boldsymbol{u}_{\mathrm{dq}}(t) - R_{\mathrm{s}}\boldsymbol{i}_{\mathrm{dq}}(t) - p\omega_{\mathrm{me}}(t)\boldsymbol{J}\boldsymbol{\psi}_{\mathrm{dq}}(t)$$

$$\boldsymbol{\psi}_{\mathrm{dq}}(t) = \begin{bmatrix} \psi_{\mathrm{d}}(t) \\ \psi_{\mathrm{q}}(t) \end{bmatrix}, \boldsymbol{i}_{\mathrm{dq}}(t) = \begin{bmatrix} i_{\mathrm{d}}(t) \\ i_{\mathrm{q}}(t) \end{bmatrix}, \boldsymbol{u}_{\mathrm{dq}}(t) = \begin{bmatrix} u_{\mathrm{d}}(t) \\ u_{\mathrm{q}}(t) \end{bmatrix}$$

$$\boldsymbol{J} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \quad i_{\mathrm{s}}(t) = \|\boldsymbol{i}_{\mathrm{dq}}(t)\|_2$$

$$T(t) = \frac{3}{2}p\boldsymbol{i}_{\mathrm{dq}}^{\top}(t)\boldsymbol{J}\boldsymbol{\psi}_{\mathrm{dq}}(t) = \frac{3}{2}p\left(\psi_{\mathrm{d}}(t)i_{\mathrm{q}}(t) - \psi_{\mathrm{q}}(t)i_{\mathrm{d}}(t)\right) \quad (2)$$

with magnetic flux linkages $\boldsymbol{\psi}_{\mathrm{dq}}$, stator currents $\boldsymbol{i}_{\mathrm{dq}}$, stator voltages $\boldsymbol{u}_{\mathrm{dq}}$, angular velocity $\omega_{\mathrm{me}} = \frac{1}{p}\frac{\mathrm{d}}{\mathrm{d}t}\varepsilon_{\mathrm{el}}$, and generated drive torque T. Further parameters of this ordinary differential equation are the stator resistance R_{s} and the number of pole pairs p. The dependence between magnetic flux linkage and stator current follows a nonlinear but static relation

$$\boldsymbol{\psi}_{\mathrm{dq}}(t) = \boldsymbol{\psi}_{\mathrm{dq}}\left(\boldsymbol{i}_{\mathrm{dq}}(t)\right). \quad (3)$$

The presented motor model incorporates several common variants as special cases, e.g., PMSMs with both, interior (IPMSM) and surface-mounted magnets (SPMSM) as well as highly

[3]Bold lowercase letters $\boldsymbol{x}$ denote vectors and bold uppercase letters $\boldsymbol{X}$ denote matrices.

TABLE I
CORRESPONDENCE BETWEEN THE CONTROL ACTION, THE SWITCHING COMMANDS AND APPLIED VOLTAGES

a	s_a	s_b	s_c	u_a	u_b	u_c	u_α	u_β
0	−	−	−	$-\frac{u_\mathrm{DC}}{2}$	$-\frac{u_\mathrm{DC}}{2}$	$-\frac{u_\mathrm{DC}}{2}$	0 V	0 V
1	+	−	−	$+\frac{u_\mathrm{DC}}{2}$	$-\frac{u_\mathrm{DC}}{2}$	$-\frac{u_\mathrm{DC}}{2}$	$+\frac{2u_\mathrm{DC}}{3}$	0 V
2	+	+	−	$+\frac{u_\mathrm{DC}}{2}$	$+\frac{u_\mathrm{DC}}{2}$	$-\frac{u_\mathrm{DC}}{2}$	$+\frac{u_\mathrm{DC}}{3}$	$+\frac{u_\mathrm{DC}}{\sqrt{3}}$
3	−	+	−	$-\frac{u_\mathrm{DC}}{2}$	$+\frac{u_\mathrm{DC}}{2}$	$-\frac{u_\mathrm{DC}}{2}$	$-\frac{u_\mathrm{DC}}{3}$	$+\frac{u_\mathrm{DC}}{\sqrt{3}}$
4	−	+	+	$-\frac{u_\mathrm{DC}}{2}$	$+\frac{u_\mathrm{DC}}{2}$	$+\frac{u_\mathrm{DC}}{2}$	$-\frac{2u_\mathrm{DC}}{3}$	0 V
5	−	−	+	$-\frac{u_\mathrm{DC}}{2}$	$-\frac{u_\mathrm{DC}}{2}$	$+\frac{u_\mathrm{DC}}{2}$	$-\frac{u_\mathrm{DC}}{3}$	$-\frac{u_\mathrm{DC}}{\sqrt{3}}$
6	+	−	+	$+\frac{u_\mathrm{DC}}{2}$	$-\frac{u_\mathrm{DC}}{2}$	$+\frac{u_\mathrm{DC}}{2}$	$+\frac{u_\mathrm{DC}}{3}$	$-\frac{u_\mathrm{DC}}{\sqrt{3}}$
7	+	+	+	$+\frac{u_\mathrm{DC}}{2}$	$+\frac{u_\mathrm{DC}}{2}$	$+\frac{u_\mathrm{DC}}{2}$	0 V	0 V

utilized PMSMs (which feature dominant magnetic saturation behavior) and synchronous reluctance motors (SynRM), which all differ concerning the nature of (3). In the following, no system-specific information (e.g., electric parameterization or lookup table) is utilized by the RL-based control algorithm. Merely an estimation of the motor current limitation, which is an upper boundary for the stator current i_s, must be available for safety purposes. The presented motor model is used to illustrate the controlled system behavior for the reader's convenience, but is not known to the learning control algorithm in any way.

B. *Two-Level Voltage Source Inverter*

The two-level voltage source inverter is another standard component of drive systems. It is supplied by means of a DC-link voltage $u_\mathrm{DC}(t)$. In the context of finite-control-set (FCS) applications such as the proposed DQ-DTC, the inverter is operated with respect to its eight distinguishable switching states as listed in Table I. The specific switching commands for the three half bridges $(s_\mathrm{a}, s_\mathrm{b}, s_\mathrm{c})$ determine the voltage that is applied to the motor, enabling control capabilities to change the electric and mechanic operating point of the drive system.

Alternatively, the inverter could also be operated with an intermediate modulator (e.g., space vector modulation), which would enable a dynamically averaged synthesization of the commanded voltages. This allows to directly command the applied voltage $\boldsymbol{u}_\mathrm{dq}$, rendering the system a continuous-control-set (CCS) application. Control schemes of this nature are equally well represented in drive control but are not within the scope of this contribution. Hence, the term DQ-DTC refers exclusively to its FCS implementation within the scope of this article.

III. DEEP Q DIRECT TORQUE CONTROL (DQ-DTC)

This contribution presents an augmented version of the DQ-DTC, which was originally proposed in [9]. Before the fundamental concept of this control scheme is reviewed, a short definition of the optimal torque control task is delivered.

A. *Optimal Torque Control*

The task of optimal torque control of PMSMs can be described by the following discrete-time dynamic optimization problem:

$$\begin{aligned} &\min_{a_k} i_{\mathrm{s},k} \quad \forall k \\ \text{s.t.} \quad & T_k = T_k^* \\ & i_{\mathrm{s},k} \leq i_\mathrm{lim} \\ & a_k \in \{0, \ldots, 7\}. \end{aligned} \tag{4}$$

Thereby, the digital sampling index is denoted by k. This formulation demands to minimize the stator current i_s (as a proxy for maximizing the drive's efficiency [1]) while sustaining the commanded reference torque T^* by means of the applied switching state a. Meanwhile, the current limit i_lim must be respected at all times, resulting in the assumption that T^* is reachable under this condition.

The DQ-DTC approach seeks to solve this optimal control problem in a data-driven fashion, allowing the setup of an optimal controller without the need for a specific motor model, i.e., knowledge about the parameters or lookup tables concerning (2) and (3) are not utilized to design the controller. This distinguishes the data-driven control approach from the conventional procedure of the model-driven control design that is usually employed when configuring, e.g., field-oriented proportional-integral controllers or MPC schemes. Only information that is generally valid is considered, e.g., the structure of (2), the staticness of (3) or the existence of operational constraints, as described in the following.

B. *Operating Conditions*

As already defined as part of the control task (4), the most important safety constraint for usage of the PMSM is defined by the current limit i_lim. Overshooting this limit can lead to a thermal overload of the drive and the feeding inverter and must, therefore, be avoided. Operating directly below i_lim is not instantaneously harmful but should not be sustained for long intervals because of the limited overheating capacity of the motor. This motivates the definition of the nominal current $i_\mathrm{n} < i_\mathrm{lim}$, which is the maximum stator current that can be endured permanently. In most drive applications, the utilized region of operation is margined by the nominal current. In special cases, overloading may also be tolerated but corresponding applications require careful temperature monitoring, which is not in the scope of this contribution. Hence, it is targeted to operate the PMSM such that $i_\mathrm{s} \leq i_\mathrm{n}$.

Whereas the current boundaries are operational constraints, the applied voltage and the voltage limitations are subject to the installed inverter and voltage source. As already stated in Table I, the DC-link voltage u_DC is of central importance for the control behavior of the motor. Therefore, operating points can only be sustained if the necessary mean voltage is available, leading to the relation

$$\|\boldsymbol{u}_\mathrm{dq}\|_2 = \|\boldsymbol{u}_{\alpha\beta}\|_2 \leq \frac{2}{\pi} u_\mathrm{DC}. \tag{5}$$

As of (2), this is specifically critical at high speed, because the induced voltage needs to be compensated by the applied voltage and only the remaining surplus can serve as a control reserve. If no such compensation is possible, the motor can become uncontrollable, which is an indirect safety concern and should be avoided.

In addition to the aforementioned, safety-related operating conditions, efficiency-related conditions can be formulated that are universally applicable for all subclasses of PMSMs. Principally, motor operation with $i_\mathrm{d} > 0$ is not practical and should, therefore, be avoided in order to maximize motor efficiency [12].[4] FCS schemes, however, feature a larger current ripple than CCS methods and can, therefore, benefit from permitting some leeway $i_{\mathrm{d}+} > 0$ in order to reduce the average current drain.

While operating the drive at commanded torque, it is pursued to maximize the efficiency of the motor by reducing the stator current i_s, leading to operation with the well-established maximum-torque-per-current (MTPC) characteristic. Torque demands that exceed the described voltage limitations, which usually happens at high speed, necessitate maximum-torque-per-voltage (MTPV) operation. While conventional control schemes exploit plant-specific parameter knowledge to determine operating points with MTPC or MTPV characteristic [13], the DQ-DTC is able to feature these behaviors after a data-driven training phase.

C. Control Design

Similar to conventional optimal control schemes such as MPC, the behavior of the controller is mainly defined by a reward function that is to be maximized.[5] Contrary to conventional optimal control, the DQ-DTC scheme does not employ optimization in real time to solve the optimization problem (4). Instead, the control strategy is learned asynchronously during direct interaction with the plant system to find the state-action value

$$q(\boldsymbol{o}_k, a_k) = \mathrm{E}\left\{r_{k+1} + \gamma r_{k+2} + \gamma^2 r_{k+3} + \ldots | \boldsymbol{o}_k, a_k\right\}. \quad (6)$$

Herein, $\boldsymbol{o}$ denotes the observation vector, which contains the information about the system state and is additionally augmented with task-specific features that simplify the learning problem (so-called feature engineering), $\mathrm{E}\{\cdot\}$ is the expected value,[6] and a is the switching action that is to be applied. The reward function r reflects the quality of the momentary plant state and must, therefore, incorporate the objectives and boundary conditions of the control problem. The discount factor γ ensures convergence of the series and must be defined in the interval [0,1[. A large discount factor translates to consideration of long-term effects when choosing an action, whereas a smaller discount factor causes the control agent to act rather short sighted.

[4]For SynRM, operating the drive at positive or negative i_d is equally valid. Only the latter case is considered here to guarantee general applicability of the DQ-DTC.

[5]Classical control usually defines cost functions that are to be minimized. The RL domain defines reward functions that are to be maximized. The latter viewpoint is presented in this contribution to stay within the RL conventions.

[6]Note that variables in the argument of the expected value are considered random variables. A distinctive notation for random variables is avoided for ease of reading.

The controller training is finished as soon as q is found, because the optimal action can then be determined easily via

$$a_k^* = \arg\max_{a'} q(\boldsymbol{o}_k, a') \quad (7)$$

meaning that the action that scores best in terms of action value is considered optimal, which is determined by comparing the resulting action values for all possible actions.

The most well-known algorithm employed to learn q with usage of an ANN in systems with continuous state and finite action space is famously known as deep Q-network (DQN) [14], [15], giving origin to the title DQ-DTC. The fundamental cornerstones of the original DQ-DTC are revisited in the following.

To approximate the state-action value by means of an ANN $\hat{q}_{\boldsymbol{\theta}}$ with network weights $\boldsymbol{\theta}$, a cost function must be formulated to allow training/optimization of $\boldsymbol{\theta}$. First, please note that according to (6), the state-action value approximation[7] must satisfy the Bellman equation [16]

$$\hat{q}_{\boldsymbol{\theta}}(\boldsymbol{o}_k, a_k) = \mathrm{E}\{r_{k+1} + \gamma\hat{q}_{\boldsymbol{\theta}}(\boldsymbol{o}_{k+1}, a_{k+1}) | \boldsymbol{o}_k, a_k\}. \quad (8)$$

This property is exploited in many pertinent RL algorithms because it allows the formulation of a cost function J_q for optimization: the left-hand side and the right-hand side should be equivalent, which means their (quadratic) distance is to be minimized, yielding

$$J_q(\boldsymbol{\theta}) = \frac{1}{|\mathcal{B}|}\sum_{\mathcal{E}_k \in \mathcal{B}} \Bigg(\hat{q}_{\boldsymbol{\theta}}(\boldsymbol{o}_k, a_k) - \underbrace{\left(r_{k+1} + \gamma(1 - d_{k+1})\max_{a'} \hat{q}_{\boldsymbol{\theta}}(\boldsymbol{o}_{k+1}, a')\right)}_{\text{estimation target}} \Bigg)^2 . \quad (9)$$

Herein, $\mathcal{B}$ denotes a minibatch of experiences $\mathcal{E}$ as follows:

$$\mathcal{E}_k = \{\boldsymbol{o}_k, a_k, r_{k+1}, d_{k+1}, \boldsymbol{o}_{k+1}\} \quad (10)$$

which contains the relevant information about the state transition that is learned from. The Boolean flag d marks the termination of the control task, which nullifies the future value when, e.g., the control task is halted or violation of safety constraints trigger an emergency shutdown. As realized by means of the $\max(\cdot)$ operator in the estimation target, this cost function focuses the momentary action value for the assumption of subsequent optimal control, i.e., the controller learns on the basis of the best achievable expected trajectory instead of the actually observed one. This implementation detail enables off-policy training, meaning that samples can be considered in any order and from any control policy in order to optimize for $\boldsymbol{\theta}$. The parameter update is then simply performed via stochastic gradient descent (or variations thereof, most famously [17])

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta\nabla_{\boldsymbol{\theta}} J_q(\boldsymbol{\theta}) \quad (11)$$

with learning rate β.

[7]The hat notation $\hat{x}$ denotes an estimation / approximator for x.

TABLE II
REWARD DISTRIBUTION FOR THE DQ-DTC IN CONSIDERATION OF SAFEGUARD ACTIVATION

name	definition	condition	reward
$\mathbb{E}$	current limit violation	$i_\mathrm{lim} < i_{\mathrm{s},k} \rightarrow d_k = 1$	$r_k = -1$
$\mathbb{E}_\mathrm{S}$	prevented cur. lim. violation	$i_\mathrm{lim} < \hat{i}_{\mathrm{s},k}$	$r_k = -(1-\gamma)$
$\mathbb{D}$	short-time overcurrent	$i_\mathrm{n} < i_{\mathrm{s},k} < i_\mathrm{lim}$	$r_k = \left(1 - \frac{i_{\mathrm{s},k} - i_\mathrm{n}}{i_\mathrm{lim} - i_\mathrm{n}}\right) \frac{1-\gamma}{2} - (1-\gamma)$
$\mathbb{D}_\mathrm{S}$	prev. short-time overcurrent	$i_\mathrm{n} < \hat{i}_{\mathrm{s},k} < i_\mathrm{lim}$	$r_k = -\frac{1-\gamma}{2}$
$\mathbb{C}$	flux amplification operation	$i_{\mathrm{s},k} < i_\mathrm{n} \land i_\mathrm{d+} < i_{\mathrm{d},k}$	$r_k = \left(1 - \frac{i_{\mathrm{d},k} - i_\mathrm{d+}}{i_\mathrm{n} - i_\mathrm{d+}}\right) \frac{1-\gamma}{2} - \frac{1-\gamma}{2}$
$\mathbb{B}_\mathrm{S}$	prevented undervoltage	$i_{\mathrm{s},k} < i_\mathrm{n} \land i_{\mathrm{d},k} < i_\mathrm{d+} \land \frac{2}{\pi} u_\mathrm{DC} < \lvert\lvert \hat{u}_{\mathrm{dq},k+1} \rvert\rvert_2$	$r_k = 0$
$\mathbb{B}$	flux weakening operation	$i_{\mathrm{s},k} < i_\mathrm{n} \land i_{\mathrm{d},k} < i_\mathrm{d+} \land \lvert\lvert \hat{u}_{\mathrm{dq},k+1} \rvert\rvert_2 < \frac{2}{\pi} u_\mathrm{DC} \land T_\mathrm{tol} < \lvert T^*_k - T \rvert$	$r_k = \left(1 - \left\lvert \frac{T^*_k - T_k}{2 T_\mathrm{lim}} \right\rvert\right) \frac{1-\gamma}{2}$
$\mathbb{A}$	reference torque isoline	$i_{\mathrm{s},k} < i_\mathrm{n} \land i_{\mathrm{d},k} < i_\mathrm{d+} \land \lvert\lvert \hat{u}_{\mathrm{dq},k+1} \rvert\rvert_2 < \frac{2}{\pi} u_\mathrm{DC} \land \lvert T^*_k - T \rvert < T_\mathrm{tol}$	$r_k = \left(1 - \frac{i_{\mathrm{s},k}}{i_\mathrm{lim}}\right) \frac{1-\gamma}{2} + \frac{1-\gamma}{2}$

The utilization of $\hat{q}_{\boldsymbol{\theta}}$ to estimate its own estimation target is labeled a bootstrapping method. In order to reduce the variance of parameter updates, estimator $\hat{q}_{\boldsymbol{\theta}}$ and target $r + \gamma \hat{q}_{\boldsymbol{\theta}}$ are usually not updated at the same rate. Instead, a set of less-frequently- or slower-updated target parameters $\boldsymbol{\theta}_\mathrm{target}$ is used to determine the estimation target [15].

As of (6), it can be seen that the action value is determined by means of the immediate and future rewards. Hence, the definition of a proper reward function is of major importance for the performance potential of the resulting DQ-DTC agent. The considered reward function is defined in Table II and depicted in Fig. 2, for which an in-depth motivation and derivation has been delivered in [9]. Please note that the original DQ-DTC approach only considered the cases $\mathbb{A}, \mathbb{B}, \mathbb{C}, \mathbb{D}$, and $\mathbb{E}$. The newly distinguished cases $\mathbb{E}_\mathrm{S}$ and $\mathbb{D}_\mathrm{S}$ correspond to further employed safety measures, which are discussed in Section IV.

The original definition of reward from [9] handles all aforementioned operation specifications with exception of the voltage boundary, whose consideration is hardly possible without availability of any model. Therefore, a data-driven prediction model is presented in the following section. It enables the control agent to adhere to the voltage boundary more reliably without any additional requirements concerning plant-specific expert knowledge.

Finally, to enable DQN estimation of the future development of the reward as of (6), it must be equipped to perform this prediction solely on the basis of $\boldsymbol{o}_k$ and a_k. This means that $\boldsymbol{o}_k$ must incorporate all necessary information about the plant state that render the control agent capable of such a prediction while simultaneously avoiding redundant features. Moreover, it is a well-established best practice to normalize the input of ANNs to the range [-1,1], which is easily performed when the motor limitations are known

$$\boldsymbol{o}_k = \Bigg[\frac{\omega_{\mathrm{me},k}}{\omega_\mathrm{me,lim}} \quad \frac{\boldsymbol{i}^\top_{\mathrm{dq},k}}{i_\mathrm{lim}} \quad \frac{3\boldsymbol{u}^\top_{\mathrm{dq},k-1}}{2u_{\mathrm{DC},k}} \quad \frac{3\boldsymbol{u}^\top_{\mathrm{dq},k-2}}{2u_{\mathrm{DC},k}}$$
$$\frac{3\boldsymbol{u}^\top_{\mathrm{dq},k-3}}{2u_{\mathrm{DC},k}} \quad \cos(\varepsilon_{\mathrm{el},k}) \quad \sin(\varepsilon_{\mathrm{el},k}) \quad 2\frac{i_{\mathrm{s},k}}{i_\mathrm{lim}} - 1 \tag{12}$$
$$2\frac{u_{\mathrm{DC},k} - U_\mathrm{DC,min}}{U_\mathrm{DC,max} - U_\mathrm{DC,min}} - 1 \quad \frac{T^*_k}{T_\mathrm{lim}}\Bigg]^\top.$$

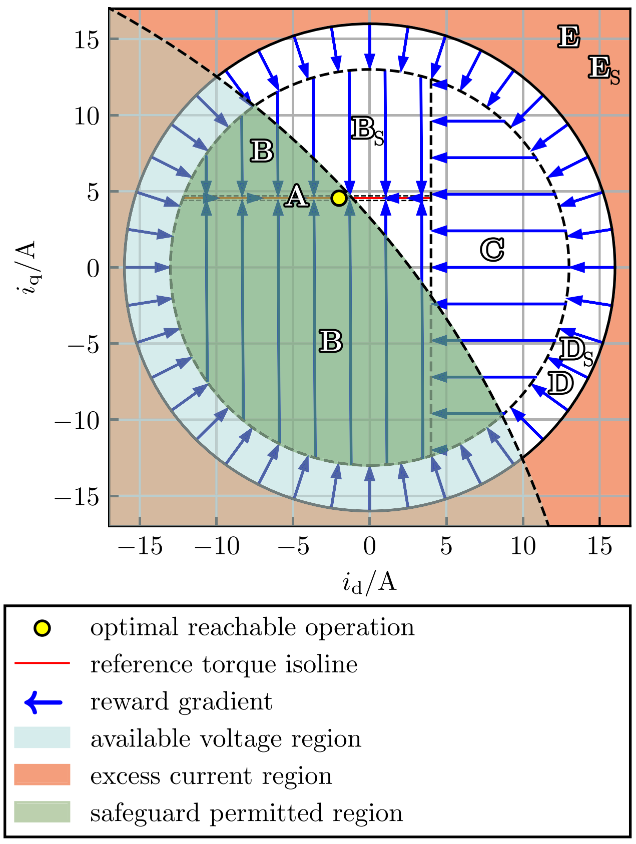


Fig. 2. Graphical depiction of the proposed reward function gradient according to Table II (oriented at [9]); please note that regions $\mathbb{B}_\mathrm{S}$, $\mathbb{D}_\mathrm{S}$, and $\mathbb{E}_\mathrm{S}$ correspond to safeguard activation and are not actually entered.

This observation design is closely related to the original proposal in [9]. The newly featured utilization of the observations $\boldsymbol{u}_{\mathrm{dq},k-2}$, $\boldsymbol{u}_{\mathrm{dq},k-3}$, and $\boldsymbol{u}_\mathrm{DC}$ stems from the real-world implementation of the DQ-DTC algorithm and is discussed more in-depth in Section V-C.

Please note that the measured torque T is considered within the reward function Table II but not within the observation vector $\boldsymbol{o}$, which means that a torque sensor is needed only during training of the DQ-DTC, but not thereafter. This is important because usual drive applications are not monitored using a torque sensor, as it introduces further cost, space, and mass demands as

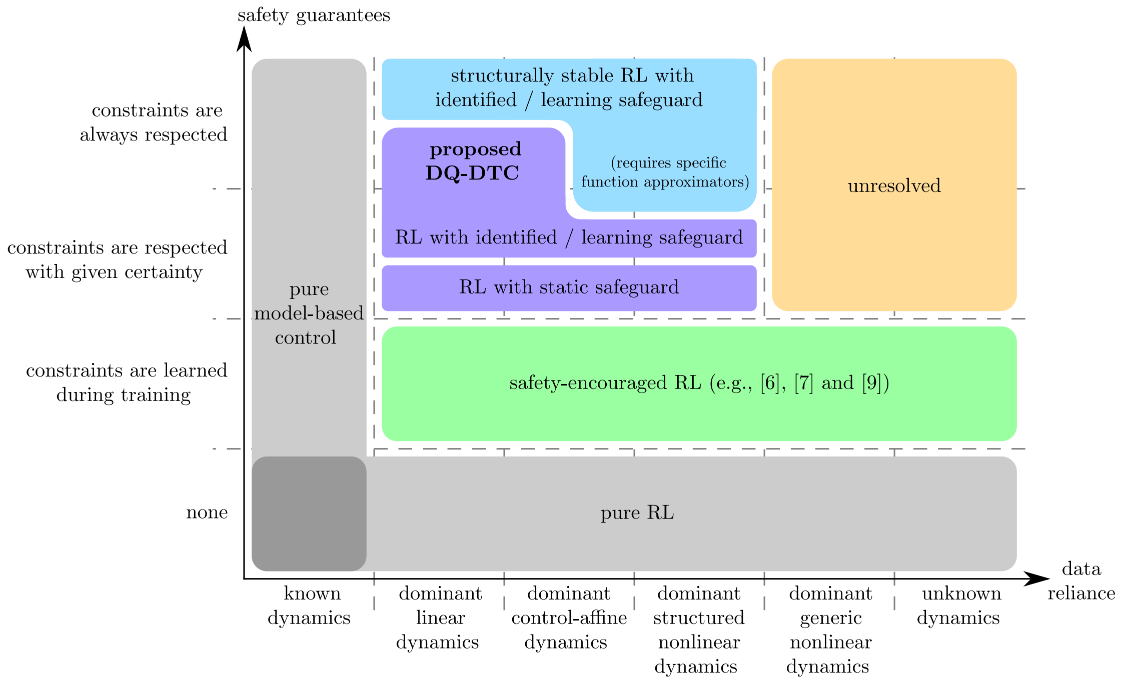

safety guarantees
constraints are always respected
constraints are respected with given certainty
constraints are learned during training
none
pure model-based control
structurally stable RL with identified / learning safeguard
(requires specific function approximators)
proposed DQ-DTC
RL with identified / learning safeguard
RL with static safeguard
unresolved
safety-encouraged RL (e.g., [6], [7] and [9])
pure RL
data reliance
known dynamics
dominant linear dynamics
dominant control-affine dynamics
dominant structured nonlinear dynamics
dominant generic nonlinear dynamics
unknown dynamics

Fig. 3. Overview of available safeguarding mechanisms and classification of the DQ-DTC (derived from [18]).

well as risk of failure. Availability of such a device is, therefore, not assumed for the utilization of the trained DQ-DTC.

IV. Safeguarding

RL algorithms can only learn through proper exploration of the state and action space, meaning that different actions may be applied in a trial-and-error fashion in all regions of the state space. Within the DQN, this is traditionally ensured through utilization of the ϵ-greedy policy[8]

$$a = \begin{cases} a^* & \text{with probability } 1-\epsilon \\ \in_{\mathrm{R}} \mathcal{A} & \text{with probability } \epsilon \end{cases} \tag{13}$$

wherein $\mathcal{A}$ denotes the available action space and $\epsilon \in [0, 1]$ is a configurable parameter of the DQN training.

This is traditionally conducted without regard to the limitations of the state space, which goes without any consequences in simulated environments. However, in this article, the deployment of an online-capable RL control agent on a real-world drive system is targeted, which incorporates safety hazards as well as limitations to experiment time.

To prevent operation in safety-critical states, this implementation employs a safeguarding routine that makes use of online system identification. This setup allows the safeguard to detect and evade the selection of unsafe actions, which keeps the point of operation within the predefined limitations. Primarily, this measure ensures safe operation of the DQ-DTC during the online training phase. Secondarily, it also accelerates the agent's training as fewer (if any) plant shutdowns with consecutive restarts occur. A classification of this safeguarding procedure is visualized in Fig. 3. The depicted categories have been motivated in [18].

A. Data-Driven Online System Identification

The employed safeguard is based on the data-driven recursive least squares (RLS) implementation for synchronous machines as presented and validated in [19]. The assumed system dynamics have the form

$$\underbrace{\begin{bmatrix} i_{\mathrm{d},k+1} \\ i_{\mathrm{q},k+1} \end{bmatrix}}_{\boldsymbol{i}_{\mathrm{dq},k+1}} = \hat{\boldsymbol{A}}_k \underbrace{\begin{bmatrix} i_{\mathrm{d},k} \\ i_{\mathrm{q},k} \end{bmatrix}}_{\boldsymbol{i}_{\mathrm{dq},k}} + \hat{\boldsymbol{B}}_k \underbrace{\begin{bmatrix} u_{\mathrm{d},k} \\ u_{\mathrm{q},k} \end{bmatrix}}_{\boldsymbol{u}_{\mathrm{dq},k}} + \hat{\boldsymbol{e}}_k \tag{14}$$

which motivates the online regression problem with regressors

$$\boldsymbol{\xi}_k = \begin{bmatrix} i_{\mathrm{d},k} & i_{\mathrm{q},k} & u_{\mathrm{d},k} & u_{\mathrm{q},k} & 1 \end{bmatrix}^\top \tag{15}$$

with measurements

$$\phi_{\mathrm{d},k} = i_{\mathrm{d},k}, \qquad \phi_{\mathrm{q},k} = i_{\mathrm{q},k} \tag{16}$$

and with parameters

$$\begin{aligned} \hat{\boldsymbol{\chi}}_{\mathrm{d},k} &= \begin{bmatrix} \hat{a}_{11,k} & \hat{a}_{12,k} & \hat{b}_{11,k} & \hat{b}_{12,k} & \hat{e}_{1,k} \end{bmatrix}^\top \\ \hat{\boldsymbol{\chi}}_{\mathrm{q},k} &= \begin{bmatrix} \hat{a}_{21,k} & \hat{a}_{22,k} & \hat{b}_{21,k} & \hat{b}_{22,k} & \hat{e}_{2,k} \end{bmatrix}^\top. \end{aligned} \tag{17}$$

Herein, the entries of the parameter vectors $\hat{\boldsymbol{\chi}}_{\mathrm{d}}$ and $\hat{\boldsymbol{\chi}}_{\mathrm{q}}$ correspond to the entries of the matrices $\hat{\boldsymbol{A}}$, $\hat{\boldsymbol{B}}$, and $\hat{\boldsymbol{e}}$, respectively.

[8]The notation style ϵ denotes the randomization threshold of the DQN, while $\varepsilon_{\mathrm{el}}$ denotes the electric motor angle. The notation $\in_{\mathrm{R}}$ refers uniform random sampling from the specified set.

Note that this system model is not linked to the physical parameters of the motor but is implemented in a purely data-driven fashion. Determination of the physical plant system parameters is, therefore, not targeted. Additional feature engineering, i.e., extending the regressor and parameter sets, can be optionally conducted if the aforementioned data-driven model structure is not leading to satisfactory results (e.g., using [20]).

The regression problem can then be tackled using the standard RLS algorithm

$$\begin{aligned}\boldsymbol{\eta}_{k-1} &= \frac{\boldsymbol{P}_{k-1}\boldsymbol{\xi}_{k-1}}{\lambda + \boldsymbol{\xi}_{k-1}^\top \boldsymbol{P}_{k-1}\boldsymbol{\xi}_{k-1}}\\ \hat{\boldsymbol{\chi}}_k &= \hat{\boldsymbol{\chi}}_{k-1} + \boldsymbol{\eta}_{k-1}\left(\phi_k - \boldsymbol{\xi}_{k-1}^\top \hat{\boldsymbol{\chi}}_{k-1}\right)\\ \boldsymbol{P}_k &= \frac{1}{\lambda}\left(\boldsymbol{I} - \boldsymbol{\eta}_{k-1}\boldsymbol{\xi}_{k-1}^\top\right)\boldsymbol{P}_{k-1}.\end{aligned} \tag{18}$$

Here, $\boldsymbol{P}$ is a scaled covariance matrix of the parameter estimation $\mathrm{Cov}(\hat{\boldsymbol{\chi}}, \hat{\boldsymbol{\chi}})$, $\boldsymbol{I}$ is the identity matrix, and $\lambda \in]0, 1[$ denotes the forgetting factor, which is a tuning parameter that defines the priority given to the regressors measured in the more distant past. Please note that the update of $\hat{\boldsymbol{\chi}}$ needs to be performed for both, the *d*-axis and the *q*-axis. An initial value for the parameter covariance $\boldsymbol{P}_0$ as well as an initial guess for the parameter vectors $\hat{\boldsymbol{\chi}}_{\mathrm{d},0}, \hat{\boldsymbol{\chi}}_{\mathrm{q},0}$ needs to be specified to run the algorithm. These initial guesses allow incorporation of expert knowledge. To underline independence of such, this contribution utilizes

$$\hat{\boldsymbol{\chi}}_{\mathrm{d},0} = \hat{\boldsymbol{\chi}}_{\mathrm{q},0} = \begin{bmatrix}0 & 0 & 0 & 0 & 1\end{bmatrix}^\top. \tag{19}$$

FCS schemes, including the presented approach, incorporate sufficient system excitation that usually allow identification with satisfying accuracy at all times.

With these considerations, the parameters $\hat{\boldsymbol{A}}$, $\hat{\boldsymbol{B}}$, and $\hat{\boldsymbol{e}}$ of the assumed plant model (14) can be considered as known, which allows to predict the outcome of the different applicable switching states on the plant. In MPC, such an identified prediction model could be utilized to determine the optimal possible switching action in consideration of a limited prediction horizon. Especially FCS-MPC suffers from exponentially increasing computational complexity with rising prediction horizon, and hence, it is typically implemented with only narrow foresight.

Note that also model-free predictive controllers (e.g., [19], [21], or [22]) suffer from the characteristic of growing computational effort. Herein, a predictive current model is identified online using the proposed RLS or a comparably adequate method. Commonly, the online identification is limited to the electric subsystem's behavior, because industrial drive applications do not incorporate a torque sensor by default, which would be required for identifying the mechanic subsystem. Therefore, also the capability of model-free predictive approaches is usually limited to current control (which is an intermediate step when designing a torque controller), whereas the proposed DQ-DTC is a direct torque controller, and hence, does not require manual design of an underlying current controller.

Contrary to predictive controllers, the DQN $\hat{q}_{\boldsymbol{\theta}}$ considers also the long-term control consequences by means of the Bellman equation, whereas the time horizon and the computational effort are independent of each other. The proposed solution in this article combines a computationally cheap one-step prediction to avoid unsafe actions with the benefits of a long-term-oriented control policy enabled by RL. Therefore, the prediction is not utilized in search of the optimal action, but only to exclude switching actions that would result in clearly unsafe plant operation. The detection of such actions is presented in the following.

B. Prevention of Unsafe Actions

As already mentioned in Section III-B, the critical operation limitations for the motor are specified by means of the current boundary, which is the most important safety condition, and the voltage boundary, which ensures controllability, and hence, ensures safety indirectly. Utilizing the parameters identified by the RLS, the system (14) can be used to check adherence to both of these boundaries for the upcoming state.

Concerning the current boundary, this can be done easily by verifying that the predicted current $\hat{\boldsymbol{i}}_{\mathrm{dq},k+1}$ adheres to the nominal current of the plant

$$\|\hat{\boldsymbol{i}}_{\mathrm{dq},k+1}\|_2 \leq i_{\mathrm{n}}. \tag{20}$$

To also ensure compliance with the voltage boundary (5), it must be examined whether the predicted operating point $\hat{\boldsymbol{i}}_{\mathrm{dq},k+1}$ can be sustained with the available DC-link voltage u_{DC}, resulting in the condition

$$\|\underbrace{\hat{\boldsymbol{B}}_k^{-1}\left((\boldsymbol{I} - \hat{\boldsymbol{A}}_k)\hat{\boldsymbol{i}}_{\mathrm{dq},k+1} - \hat{\boldsymbol{e}}_k\right)}_{\hat{\boldsymbol{u}}_{\mathrm{dq},k+1}}\|_2 \leq \frac{2}{\pi}u_{\mathrm{DC},k}. \tag{21}$$

Here, the expression on the left-hand side corresponds to the fundamental (average) voltage amplitude that is necessary to maintain the operating point at $\hat{\boldsymbol{i}}_{\mathrm{dq},k+1}$, i.e., steady-state control operation, while assuming that the DC-link voltage will not change drastically from one sampling instant to the next [23]. Naturally, inspection of these conditions must succeed for all switching states $a \in \mathcal{A}$ that can lead to different follow-up states $\hat{\boldsymbol{i}}^a_{\mathrm{dq},k+1}$. Each switching state that violates any of the constraints may not be selected in the momentary time step, neither in case of an exploiting nor in case of an exploring action. Under these circumstances, the safeguard overwrites the agent's policy entirely to ensure safe plant operation. An algorithmic representation of the outlined safeguarding technique is given in Algorithm 1. Herein, the prediction model $\hat{\boldsymbol{A}}, \hat{\boldsymbol{B}}, \hat{\boldsymbol{e}}$ is updated in an online fashion, without the requirement (but surely with the possibility) to insert an expert-based initial model before the training.

Please note that, in order to allow the DQ-DTC agent to learn from mistakes, the naive action a must be defined for the experience tuple $\mathcal{E}$ (10) even though only the safeguarded action a_{S} is applied. If a_{S} would be included in $\mathcal{E}$ instead, $\mathcal{E}$ would resemble a state transition wherein the potentially unsafe a is masked, consequently preventing the agent from learning to avoid such behavior.

An appropriate safeguard prevents the plant system from being operated in the regions $\mathbb{D}$ and $\mathbb{E}$ (cf., Table II and Fig. 2), which avoids emergency system shutdowns, especially in the

Algorithm 1: Safeguarded DQ-DTC.

```
for given weights $\boldsymbol{\theta}$ of $\hat{q}_{\boldsymbol{\theta}}(\boldsymbol{o}, a)$
$k \leftarrow 0$
loop
  obtain $\boldsymbol{o}_k$
  update $\hat{\boldsymbol{A}}_k$, $\hat{\boldsymbol{B}}_k$ and $\hat{\boldsymbol{e}}_k$ using the RLS (18)
  $\mathcal{A}_{\mathrm{S},k} \leftarrow \{\}$
  for all $a' \in \mathcal{A}$ do
    predict $\hat{\boldsymbol{i}}^{a'}_{\mathrm{dq},k+1}$ according to (14)
    if (20) and (21) are satisfied then
      add $a'$ to the set of safe actions $\mathcal{A}_{\mathrm{S},k}$
    end if
  end for
  if $\mathcal{U}_{[0,1]} < \epsilon$ then[9]                          ▷ explore
    obtain $a_k \in_{\mathrm{R}} \mathcal{A}$
    if $a_k \in \mathcal{A}_{\mathrm{S},k}$ then
      $a_{\mathrm{S},k} \leftarrow a_k$
    else
      obtain $a_{\mathrm{S},k} \in_{\mathrm{R}} \mathcal{A}_{\mathrm{S},k}$
    end if
  else                                                    ▷ exploit
    obtain $a_k \leftarrow \arg\max_{a' \in \mathcal{A}} \hat{q}_{\boldsymbol{\theta}}(\boldsymbol{o}_k, a')$
    obtain $a_{\mathrm{S},k} \leftarrow \arg\max_{a' \in \mathcal{A}_{\mathrm{S},k}} \hat{q}_{\boldsymbol{\theta}}(\boldsymbol{o}_k, a')$
  end if
  execute $a_{\mathrm{S},k}$
  $k \leftarrow k + 1$
end loop
```

training phase when the control agent has not yet learned a feasible policy. On the other hand, the agent is unable to learn from failures if such failures do not occur. By means of the safeguard, however, prediction results that forecast unsafe switching are available and can be considered within the reward function, adding a few more cases to the reward distribution as listed in Table II and visualized in Fig. 2. The newly added case distinctions $\mathbb{B}_\mathrm{S}$, $\mathbb{D}_\mathrm{S}$, and $\mathbb{E}_\mathrm{S}$ are rewarded in such a way that triggering the safeguard is always better than actually violating the limitations. Otherwise, the agent could be encouraged to actively bypass the safeguard whenever operated in states where the prediction model (14) has problems to accurately forecast such violations.[9]

In terms of published safeguarding routines, the proposed approach is closely related to so-called *postposed shielding* [24], which is characterized by perpetual determination of safe alternative action(s) a_S and also ensures safeguarding during the application phase (i.e., even after the training phase is finished). Instead of attempting to learn the necessary shielding behavior from constraint violations, the concept of *predictive safety certification* [25] is combined with *recursive feasibility* [26] within the assumed system dynamics (14). For the given motor setup, this is done effectively and free of specific system knowledge by exploiting the discussed voltage limitation as controllability condition (5), (21).

Finally, the DQ-DTC agent is able to learn optimized operation behavior by means of measured real-world data while the consequences of safety-critical actions are predicted using the data-driven model. Actual terminations of plant operation are, therefore, obsolete and the training can be continued without any time loss or harm to the system. A schematic overview of the employed control structure is presented in Fig. 1.

V. Implementation

The real-time capable implementation of RL-based drive torque control faces some challenges that are discussed in this section. Primarily, this concerns the setup of an asynchronous edge-computing pipeline for the training, the hardware implementation of the DQ-DTC agent under consideration of real-time constraints, and the handling of parasitic effects that arise from the mechanical subsystem.

The former is necessary because a complete implementation of the RL learning algorithms as well as its inference on contemporarily available embedded computing hardware is not yet technically possible in hard real time due to the high necessary update frequency of the FCS drive control. Therefore, the learning algorithms are outsourced to more powerful computing hardware. This is implemented in the present case by means of edge computing, but in principle should also be feasible by means of cloud computing. This distributed approach turns the drive control into an Internet of Things application. Please note that all subsequently depicted time-series plots have been gathered in real-world test bench experiments, and that software-in-the-loop and hardware-in-the-loop simulations are not presented for clarity (cf., the last step of the RL controller deployment pipeline as proposed in [7]).

A. EdgeRL Pipeline

A major part of this contribution features the setup of an asynchronous edge-computing pipeline that is to be utilized during the training phase of the DQ-DTC agent. A schematic overview of the employed structure is presented in Fig. 4. As depicted, a dSPACE MicroLabBox system is in use as rapid control prototyping hardware (RCPH). From there, measured experiences $\mathcal{E}_k$ can be read in real time with the use of a corresponding Python interface, allowing time-efficient data capturing at the test bench. The collected data are sent to a workstation computer using a TCP/IP-based communication channel to guarantee causal integrity of the accrued data.

The workstation computer runs the RL training algorithm on the basis of the buffered experiences. This means that only the backward pass / gradient descent optimization of the DQN training needs to be considered here. Therefore, the training of the DQ-DTC is outsourced from the RCPH and only the forward pass / inference of the utilized ANN must be performed with real-time capability on the dSPACE system.

The newly computed DQN weights $\boldsymbol{\theta}$ are continuously sent back to the test bench computer to allow their appliance within

[9] $\mathcal{U}_{[a,b]}$ denotes a random sample from the uniform distribution on the interval $[a, b]$.

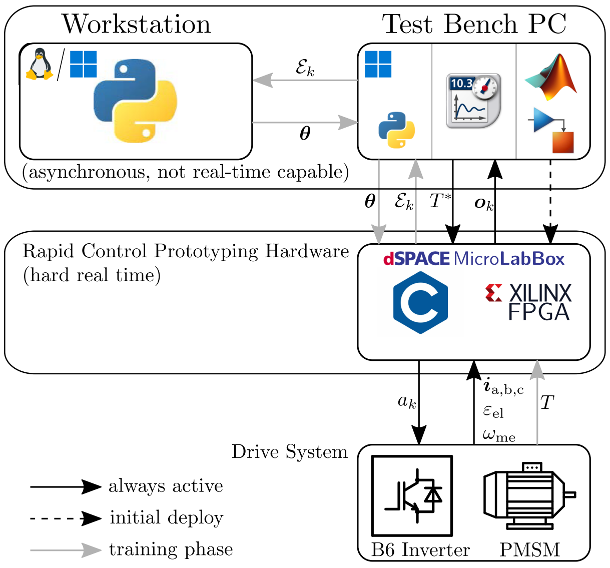


Fig. 4. Schematic of the online EdgeRL pipeline.

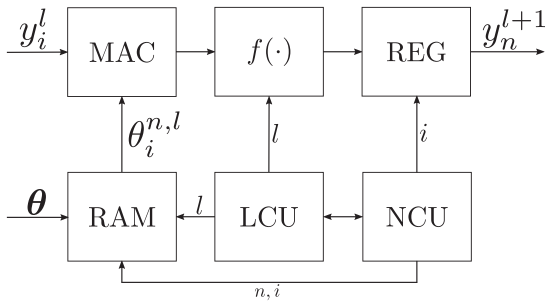


Fig. 5. Schematic of the realization of an online-configurable neuron on the FPGA, oriented at [5], components are: multiply accumulator (MAC), activation function ($f(\cdot)$), output register (REG), weight storage (RAM), layer control unit (LCU), and neuron control unit (NCU), n and l are respective identifiers of the active neuron and layer, i indicates the entries of the neuron's weight vector $\boldsymbol{\theta}^{n,l}$, and y is an (intermediate) result.

the operated control loop. To maximize throughput on the workstation end of the pipeline, the unpacking of received sample data into the training memory, the backward pass that utilizes the training memory and the dispatch of the network weights are all separated into different processes. This avoids idle time that would otherwise occur, e.g., because the training would be halted until a message is completely sent or read. Finally, also the TCP/IP communication is separated between two channels, allowing both PCs to send and receive data simultaneously, which enables the asynchrony of the tasks. This is exceptionally important in the proposed setup, because the sampling time T_{s} is much smaller than the time that is necessary for one learning step T_{l}, which means that new samples accrue much faster than network weight updates. The complete communication and training code is available at [11].

B. DQN Inference

The employed MicroLabBox system by dSPACE permits inference of the DQN (i.e., function evaluation of $\hat{q}_{\boldsymbol{\theta}}(\boldsymbol{o}_k, a_k)$) either by means of its CPU or by utilization of the built-in FPGA. As indicated in [9], sensible ANNs for the given task grow to a size where implementation of the ANN inference on the CPU is not feasible anymore to yield a result within the given sampling time of T_{s}. Here, it is more reasonable to make use of the FPGA's parallelization potential to allow adherence to the real-time constraint. In order to operate the pipeline depicted in Fig. 4, it is furthermore required that the DQN parameters $\boldsymbol{\theta}$ can be updated at runtime. A schematic of a corresponding neuron implementation is depicted in Fig. 5. As can be seen, the same hardware is utilized for all consecutive layers, allowing for a resource-efficient FPGA build. With the given structure, a DQ-DTC can be evaluated with a time demand of

$$T_{\mathrm{ANN}} = [(l-1)(n_{\mathrm{h}} + \tau_{\mathrm{n}}) + \dim(\boldsymbol{o}) + |\mathcal{A}| - 1] \cdot T_{\mathrm{FPGA}} \quad (22)$$

TABLE III
PARAMETERIZATION OF THE DQN

symbol	description	value
l_{h}	number of hidden layers	10
n_{h}	number of neurons per hidden layer	90
f_{h}	hidden layer activation function	LeakyReLU (α=0.3)
f_{out}	output layer activation function	identity
T_{FPGA}	FPGA base cycle time	10 ns
τ_{n}	neuron runtime delay	7
T_{ANN}	DQN evaluation time on FPGA	8.94 µs
T_{nu}	time demand for one neuron update	350 µs
T_{Nu}	time demand for a full network update	315 ms
T_{l}	time demand for one gradient descent	$\approx$ 10 ms

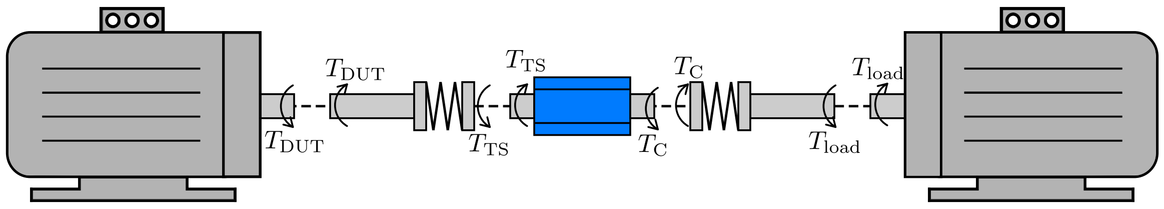
wherein l is the number of layers, n_{h} is the number of neurons per hidden layer, τ_{n} is a runtime delay each neuron needs to finish its computation, and T_{FPGA} is the base cycle time of the utilized FPGA. This configuration is sufficiently fast to conform the sampling time constraint as long as architectures do not get too comprehensive. Due to hardware constraints, reconfiguration of all network parameters $\boldsymbol{\theta}$ cannot be performed within one sampling period. Since feasible learning rates, and therefore, also the parameter change per update $\Delta\boldsymbol{\theta}$ is rather small, and because the DQN gradient descent is rather time intensive, there is no further concern to update the DQN at a larger time rate than the controller clock period T_{s}. Hence, the parameters are updated for one neuron at a time and at a slower update rate of T_{nu}, resulting in a total network update time of

$$T_{\mathrm{Nu}} = l \cdot n_{\mathrm{h}} \cdot T_{\mathrm{nu}}. \quad (23)$$

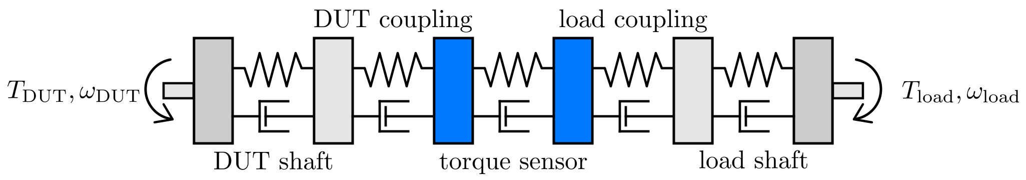
An overview of the utilized network architecture and its computational real-time demand is provided in Table III.

C. Parasitic Effects During Test Bench Training

The real-world behavior of the described motor setup does, naturally, not fit the ideal assumptions that have been featured in [9]. The rotational vibration capability of the drive train and the limited bandwidth of common torque sensors complicate



(a)


DUT coupling
load coupling
$T_{\text{DUT}}, \omega_{\text{DUT}}$
$T_{\text{load}}, \omega_{\text{load}}$
DUT shaft
torque sensor
load shaft

(b)

Fig. 6. Depiction of the mechanical drive train. (a) Free body diagram of the drive train. (b) Equivalent scheme of the drive train.

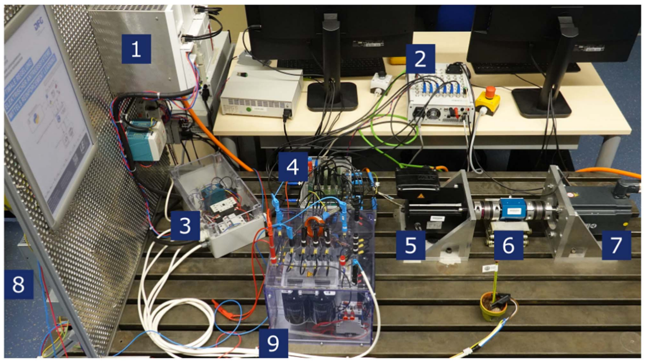


Fig. 7. Experimental test bench setup: 1) load inverter, 2) RCPH, 3) protective switch and auxiliary power supply, 4) electric sensors, 5) device under test (DUT), 6) drive train and torque sensor (without safety cover), 7) load motor, 8) DC-link chopper, and 9) DUT inverter.

the learning task even further and must be dealt with to enable model-independent learning. Moreover, the DC-link voltage u_{DC} cannot be assumed constant.

The latter issue can be dealt with rather easily. A time-variant u_{DC} can be interpreted as a state of the plant system rather than a (constant) parameter, and its consideration within the observation vector $\boldsymbol{o}$ as of (12) enables the control agent to react to corresponding changes. Since low-bandwidth voltage measurement can be realized rather inexpensive, availability of u_{DC} is the usual case in drive systems, and hence, adding it to $\boldsymbol{o}$ complicates neither the hardware nor the software setup.

Contrary to the original (purely simulative) scenario in [9], the mechanical behavior of the application adds further dynamics to the plant. A free body diagram and equivalent scheme of the drive train are depicted in Fig. 6, strongly indicating oscillating behavior that impacts the torque measurement signal, which is used during controller training. Unlike the DC-link voltage, the internal states of the drive train, i.e., angular velocities and torsional angles of each individual component named in Fig. 6(b), are not available and cannot be determined with reasonable effort, rendering them hidden states. Therefore, a different approach is suggested. To enable the DQ-DTC to correctly estimate $\hat{q}_{\boldsymbol{\theta}}$, earlier samples of the commanded voltage $\boldsymbol{u}_{\text{dq},k-1}, \boldsymbol{u}_{\text{dq},k-2}, \boldsymbol{u}_{\text{dq},k-3}$ are integrated into the observation vector $\boldsymbol{o}_k$ as of (12) as so-called lagging features [27]. Due to the correspondence between applied voltage and switching action, $\boldsymbol{o}_k$ is in this way enriched to allow the DQ-DTC to find the mapping between past actions, hidden mechanical states, and corresponding action value $\hat{q}_{\boldsymbol{\theta}}$. Hence, the direct dependence upon the drive train's state is replaced with available signals and the issues of the mechanical subsystem are handled with very manageable effort. The specific necessity of at least three lacking features has been empirically determined and a larger number did not prove beneficial. In fact, the DQ-DTC agent was unable to learn feasible control behavior without this modification.

As of [9], the consideration of at least the latest voltage command $\boldsymbol{u}_{\text{dq},k-1}$ is also necessary to allow compensability of the digital control delay of one sampling instant. The commanded switching action a_{k-1} does not show its effect in measurement $\boldsymbol{o}_k$ but rather in $\boldsymbol{o}_{k+1}$, because the computation of a_{k-1} on the basis of the measurement $\boldsymbol{o}_{k-1}$ is not instantaneously available. It is, therefore, applied starting at time k and has, hence, a direct effect on the action value $q(\boldsymbol{o}_k, a_k) = \mathrm{E}\{r_{k+1} + \gamma q(\boldsymbol{o}_{k+1}, a_{k+1}) | \boldsymbol{o}_k, a_k\}$.

VI. Experimental Results

In order to verify the feasibility of the proposed DQ-DTC approach, an experimental investigation is performed on a PMSM test bench system, which is depicted in Fig. 7. The utilized components are listed in Table IV, nameplate data of the PMSM under test and parameterization of the DQ-DTC are specified in Table V. The validation of the DQ-DTC is conducted in three steps. First, the safeguard is tested concerning its functionality. Second, the training phase for the RL agent is analyzed in terms of physical and numerical stability. Finally, the performance of the DQ-DTC is presented in a series of exemplary test scenarios. while exploration actions are deactivated.

TABLE IV
COMPONENTS OF THE TEST BENCH SYSTEM

device	identifier	manufacturer
RCPH	MicroLabBox (built-in FPGA)	dSPACE
FPGA	Kintex-7 XC7K325T FPGA	Xilinx
DUT	CM3C80S	SEW EURODRIVE
DUT inverter	SEMITEACH B6U+E1CI	SEMIKRON
load motor	AM8062-0RHA-0000	Beckhoff Automation
load inverter	AX5125-0000-0202	Beckhoff Automation
torque sensor	T8-20NM	interfaceforce

TABLE V
NOMINAL PARAMETERIZATION OF THE CONSIDERED DRIVE SYSTEM SEW CM3C80S (DERIVED FROM NAMEPLATE DATA) AND CONFIGURATION OF THE DQ-DTC WITH DEFINITION OF THE PERMITTED OPERATION REGIONS

symbol	description	value	
p	pole-pair number	4	
R_s	stator resistance	203	mΩ
L_d	d inductance	1.44	mH
L_q	q inductance	1.44	mH
ψ_p	permanent magnet flux	112	mV · s
U_DC	configured DC-link voltage	50	V
T_s	sampling time	50	µs
i_n	nominal current	13	A
i_lim	maximum current	16	A
$i_\mathrm{d+}$	tolerable positive d current	4	A
$U_\mathrm{DC,min}$	minimum DC-link voltage	25	V
$U_\mathrm{DC,max}$	maximum DC-link voltage	75	V
$n_\mathrm{me,lim}$	maximum speed[10]	750	min^{-1}
T_lim	maximum torque	10.5	N · m
T_tol	torque control tolerance	0.1	N · m

A. Safeguard Functionality

To ensure safety of components and personnel, and secondarily, to avoid training downtimes, it is of high priority to validate the functionality of the safeguarding procedure that has been proposed in Section IV. This investigation is performed for two scenarios: at low speed, where the induced voltage plays no significant role, and secondly, at higher speed where the voltage boundary is of importance. The tests are performed by presenting arbitrary, random switching actions to the safeguard algorithm, which then should be able to filter out the unsafe actions to keep the motor current trajectory $\boldsymbol{i}_\mathrm{dq}$ within the current and voltage limitations.

Long-term current trajectories for both scenarios are depicted in Fig. 8. Please note that the test bench includes a load machine as well as open-loop controlled DC-link choppers, whose dynamics are not included in the identified model (14), and therefore, cannot be considered by the safeguard. Still, the current boundary has visibly been avoided with no striking violation. Further, the identified voltage boundary is displayed with the identified voltage limit being adhered to rather consistently.

[10]Speed and angular velocity correspond via $\omega_\mathrm{me} = n_\mathrm{me}\frac{2\pi}{60}\frac{\mathrm{min}}{\mathrm{s}}$.

TABLE VI
STATISTICAL EVALUATION OF THE CURRENT PREDICTION ERROR $e_\mathrm{d,q}$ WITH MEAN μ AND STANDARD DEVIATION σ

	μ	σ
$e_{\mathrm{d},k} = \hat{i}_{\mathrm{d},k} - i_{\mathrm{d},k}$	$\mu_{e_\mathrm{d}} = 4.085 \cdot 10^{-3}\,\mathrm{A}$	$\sigma_{e_\mathrm{d}} = 4.675 \cdot 10^{-1}\,\mathrm{A}$
$e_{\mathrm{q},k} = \hat{i}_{\mathrm{q},k} - i_{\mathrm{q},k}$	$\mu_{e_\mathrm{q}} = -7.174 \cdot 10^{-6}\,\mathrm{A}$	$\sigma_{e_\mathrm{q}} = 5.239 \cdot 10^{-1}\,\mathrm{A}$

TABLE VII
TRAINING CONFIGURATION OF THE DQ-DTC

configuration parameter	specification
torque reference range	$[-6.5\,\mathrm{N\cdot m}, 6.5\,\mathrm{N\cdot m}]$
torque reference change probability	10^{-4}
speed range	$[-0.9\,n_\mathrm{me,lim}, 0.9\,n_\mathrm{me,lim}]$
speed change probability	$5 \cdot 10^{-6}$
maximum angular acceleration	$8.4\,\mathrm{s}^{-2}$
ϵ-greedy initial and final value	0.3, 0.0
learning rate initial and final value	$10^{-3}, 10^{-7}$
discount factor γ	0.85
target network update parameter	0.2
memory buffer size	$4 \cdot 10^5$
batch size $\lvert\mathcal{B}\rvert$	32
training duration	10 min

A statistical evaluation of the safeguard's prediction uncertainty is delivered in Table VI, which indicates that the minor violations, which are mainly visible in Fig. 8(b), can be attributed to the prediction uncertainty that remained during usage of the RLS from [19]. This result encourages the proposed safeguard architecture that allows certain prediction error in correspondence to the leeway between i_n and i_lim.

Overall, the functionality of the safeguard can consequently be confirmed. In fact, no violation of the current limit occurred, and hence, no emergency shutdown was necessary for the entirety of investigations in this article.

B. Training Phase

After asserting the safeguard functionality, the training phase is investigated in detail. To analyze the numerical convergence characteristic of the training phase, ten separate DQ-DTC agents have been trained independently for 10 min each, i.e., with reinitialization of a new set of random network weights for each individual agent. During the training, the torque references and the operated motor speed that is enforced by the load machine are resampled uniformly at random sampling instants as specified in Table VII. The learning rate β and the ϵ-greedy parameter are linearly decreased from the beginning to completion of the training phase (scheduling).

Several snapshots from one exemplary training phase are depicted in Fig. 9. As can be seen, the reference torque T^* and the speed n_me are varying over the training time. While the early performance looks quite insufficient due to the mostly untrained control agent, the performance at the end of the training phase is a lot more satisfying. Please note that the explorative policy

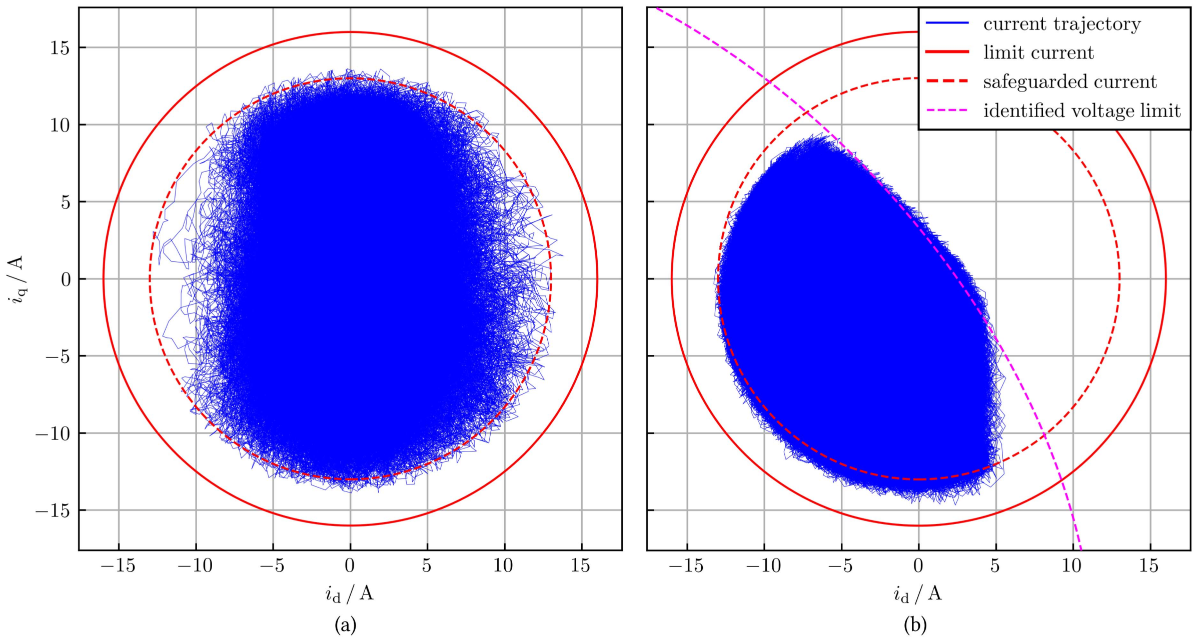


Fig. 8. Long-term current trajectory recordings to validate the safeguard's functionality. (a) Operation at low speed $|n_{me}| < 50\,\text{min}^{-1}$; only the current boundary is active. (b) Operation at high speed $n_{me} = 700\,\text{min}^{-1}$; current and voltage boundaries are active.

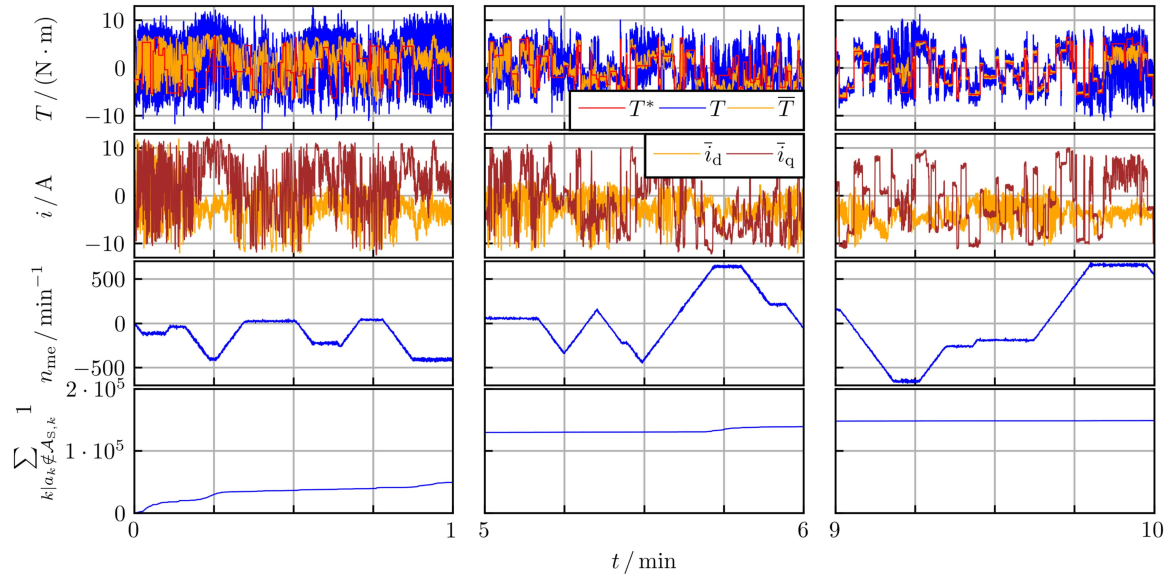


Fig. 9. Control performance in an early (left), intermediate (center), and late (right) phase of an exemplary training.

($\epsilon > 0$) is followed for the whole training time, and therefore, even the latest depicted interval features suboptimal exploration behavior. Moreover, the rather high motor speed would necessitate operation beyond the voltage boundary, which cannot be realized and always results in tracking errors. Further, it is visible that the safeguard is activated quite often within the early stages of the training, whereas there are almost no interventions necessary close to the training finish. Both, the visibly increasing control accuracy as well as the decreasingly frequent safeguard interventions, confirm the feasibility of the DQ-DTC training.

Over the course of each training, the measured reward is recorded and a statistical evaluation of the learning behavior over all ten agents is depicted in Fig. 10. As visible, the mean reward of the agent set is converging reliably, and also the corresponding standard deviation $\sigma_{\overline{r}}$ is decreasing. Strikingly, no negative outliers have been observed during the procedure, and despite the randomness of DQN initialization and training routine, the final performance in the training has been measured to be quite similar. The peak control performance is consistently reached within less than 10 min of training, i.e., the proposed

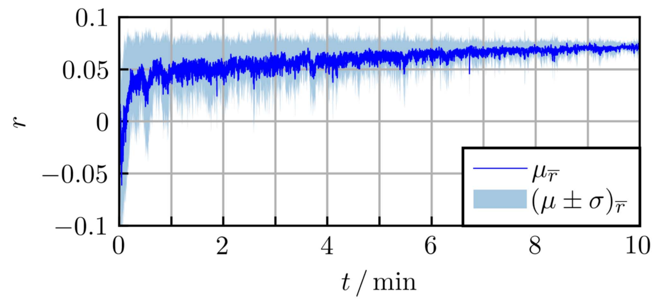


Fig. 10. Mean learning behavior $\mu_{\overline{r}}$ over ten separate DQ-DTC trainings with marked variation range of one standard deviation $\sigma_{\overline{r}}$, moving average filter applied.

RL framework learns as quick as making a pot of coffee (cf., uploaded supplementary video material [10]).

C. Performance Evaluation

Finally, to validate the DQ-DTC in practice, the performance of an exemplary trained agent is assessed in several test bench experiments, which include the following.

1) A torque reference step at constant speed.
2) A speed ramp from negative to positive velocity at constant torque reference.
3) A torque reference ramp from negative to positive torque at constant speed.
4) A small-signal investigation with several torque reference steps at constant speed.

For these tests, the best-performing agent from the previous training investigation was selected concerning its cumulative reward during training. Moving-average quantities are determined with a window size of 50 ms and are denoted by an overline ($\overline{T}, \overline{i}$).

1) Torque Reference Step: In the first experiment, a torque reference step is investigated to evaluate the control loop's reaction to transients. The obtained measurement is depicted in Fig. 11. Beside the measured torque T, also the calculated drive torque estimation $\hat{T}$ is presented, which does not feature the low-pass behavior and the oscillations of the drive train and is, hence, a more feasible basis for investigating the control behavior precisely. The computation of $\hat{T}$ is based on comprehensively identified motor characteristics, which is not compatible with the premise of crafting a control scheme without a priori knowledge and is, therefore, only used for the given validation but not within any formulation that is used within the DQ-DTC's setup or training.

Unfortunately, the given drive train features a dominant oscillatory behavior with limited bandwidth (as can be inferred from the time lag between $\hat{T}$ and T). Since the torque measurement T is utilized within the reward formulation, the parasitic drive train behavior is obviously limiting the reachable torque tracking precision of the control loop. Moreover, the current and torque ripple that is inherent to FCS approaches leads to consistent excitation of the oscillation such that a true steady state is never reached.

Despite these complications, which are all independent of the DQ-DTC, fast torque tracking of roughly 5 ms can be observed

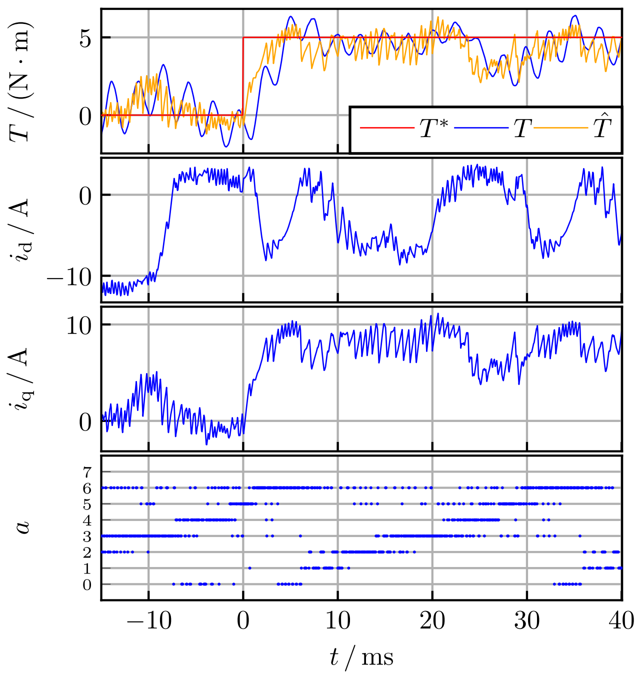


Fig. 11. Measurement of a torque reference step at $n_{\mathrm{me}} = 500\ \mathrm{min}^{-1}$, $\hat{T}$ denotes the calculated electromagnetic torque for validation purposes.

in Fig. 11, and even the time series of applied actions a features the familiar overlapping staircase form.

Interestingly, a rather large i_d was observed during the experiment. It can be speculated that this results in a decreased torque ripple, and hence, in less striking drive train oscillations, which would lead to a higher reward r.

2) Speed Ramp: A speed ramp experiment with constant torque reference is depicted in Fig. 12. Over the course of the acceleration, the torque ripple can be seen to be quite significant, which is usual for FCS control schemes (and partly also attributed to the drive train oscillation). The moving average of the torque measurement $\overline{T}$, however, features clearly that the torque reference is met for the whole considered speed range.

3) Torque Reference Ramp: Fig. 13 presents an experiment wherein the DQ-DTC is tracking a ramping reference torque. This scenario reveals no significant shortcomings. Only a small offset error can be measured when speed hits its upper steady state.

4) Small-Signal Behavior: The small-signal behavior of the DQ-DTC is featured in Fig. 14. The momentary torque measurement is omitted for clarity in this case and only the moving average $\overline{T}$ is shown. In this plot, the control agent can be observed to react with no visible delay to the changing torque reference T^*. Again, some of the commanded operating points exhibit a visible torque offset, which can presumably be attributed to their proximity to the current boundary.

Concerning all of the presented experiments, it is observed that $\overline{i}_d \neq 0$. Although such behavior seems counterintuitive when dealing with an SPMSM, several possible explanations can be identified.

1) The information that the given motor is an SPMSM has not been used for setting up the RL controller. Hence, it is not initially clear that $\overline{i}_d = 0$ should be targeted for

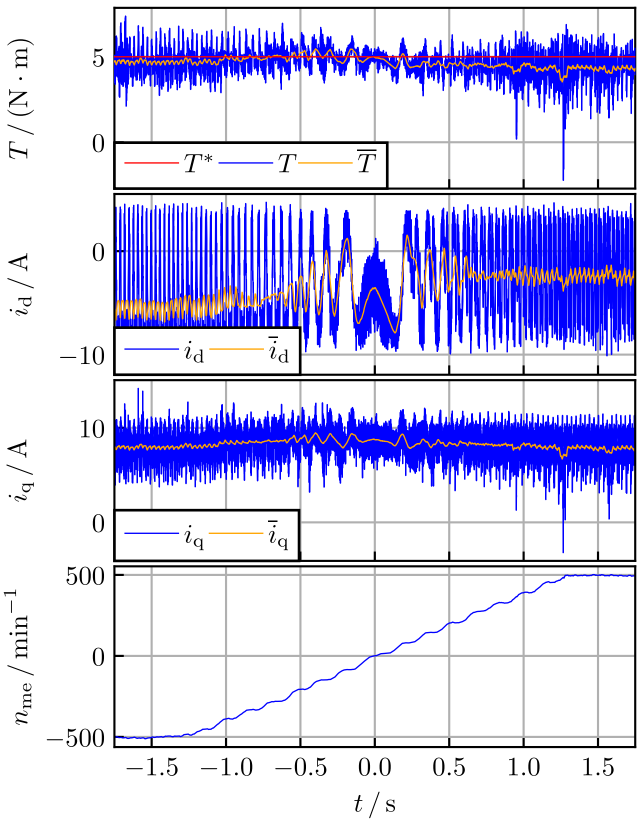


Fig. 12. Measurement of a speed ramp with additional moving-average signals.

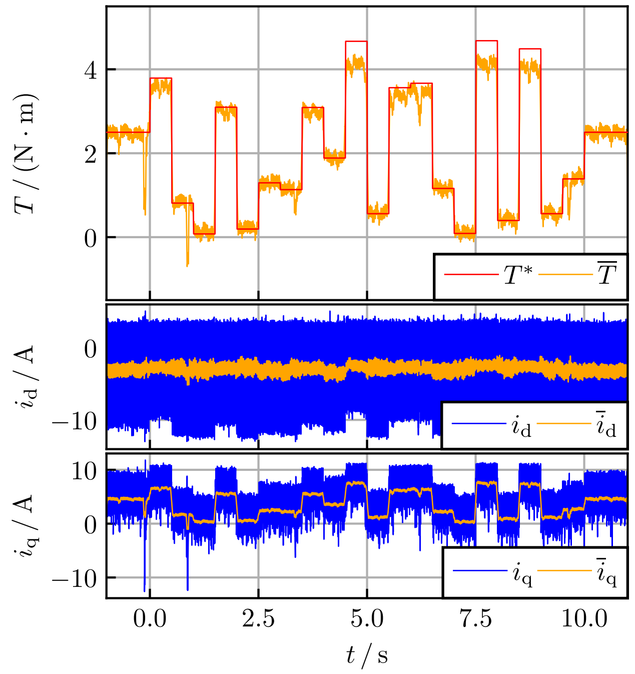


Fig. 14. Measurement of the small-signal behavior concerning the torque reference at $n_\mathrm{me} = 500\ \mathrm{min}^{-1}$ with additional moving-average signals.

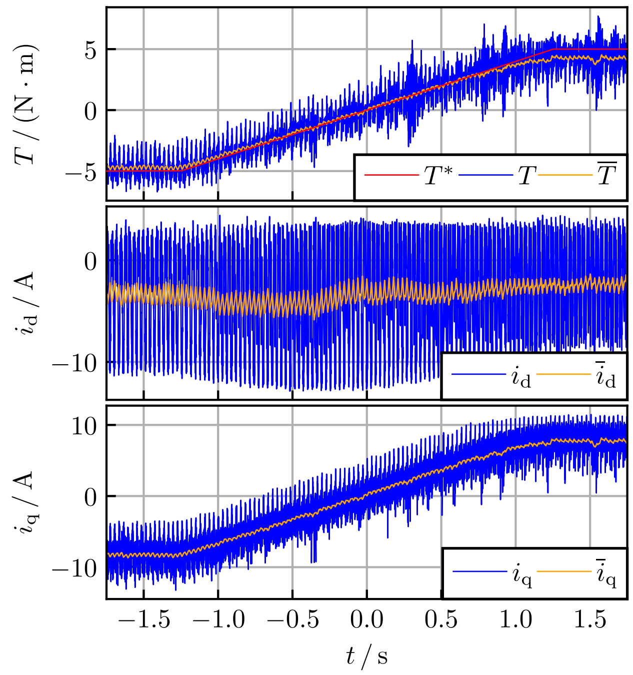


Fig. 13. Measurement of a torque reference ramp at $n_\mathrm{me} = 500\ \mathrm{min}^{-1}$ with additional moving-average signals.

maximizing efficiency, but it must be learned during the training phase.

2) The training phase might be chosen too short or might be biased in terms of flux weakening operation (provoking $\overline{i}_\mathrm{d} < 0$).
3) Numerical difference in reward may be negligible when moving $\overline{i}_\mathrm{d}$ closer to zero, which is a valid concern when dealing with the limited numerical precision of the FPGA implementation.
4) The selection of $\overline{i}_\mathrm{d} = 0$ is only optimal in terms of steady-state efficiency, which is given a lower priority than time-optimal torque tracking by means of Table II. For maximum reward, it could, therefore, be more promising to maintain $\overline{i}_\mathrm{d} < 0$, because this would allow faster transients, and correspondingly, a tracking behavior of higher bandwidth.

For the given training scenario with consecutive torque reference changes of arbitrary magnitude (cf., Fig. 9), the latter mechanism could be plausible to be the dominant reason for the observed behavior. If so, the operation with $\overline{i}_\mathrm{d} < 0$ would be exactly in line with the targeted control performance, but this needs further confirmation in future investigations.

VII. Conclusion and Outlook

A. Conclusion

The general goal of implementing and validating the DQ-DTC agent in a real-world test has been achieved. Although the initial implementation effort is significant in terms of setting up the

EdgeRL online learning pipeline and preparing the reconfigurable DQN on an FPGA, the finally resulting controller design / training expense of 10 min (and prospectively, even less) is quite low. The training process was completely automated and did not require any human intervention.

Although the reached control performance is not yet on par with the effectiveness of established optimal controllers such as MPC, the relatively short period for which RL-based control approaches have been investigated in the domain of three-phase drives promises a lot of scope for improvement and further research potential.

B. Outlook

The presented findings and insights yield a broad base for future research. Some ideas and possible directions for further investigations in RL-based power system control are listed in the following.

1) DQ-DTC Research Potential: The training behavior of the DQ-DTC featured quite fast and reliable learning. The training speed could be further improved with different types of scheduling plans for, e.g., ϵ or the learning rate β. In terms of the achievable performance, the reduction of the drive train oscillations can be expected to yield better torque precision. Besides physical manipulation of the corresponding drive setup (i.e., replacing the couplings or the torque sensor), pre- or postprocessing approaches to compensate for these oscillations would be of major interest to avoid hardware cost and manual effort. A more accurate safeguard design could be achieved by exchanging or extending the RLS-based architecture with methods that allow consideration of data-driven system identification in terms of locally different system dynamics, e.g., [28].

Concerning the drive losses, the featured reward function only takes into account the motor losses but not the inverter losses. Effective ways to limit the switching frequency would also be important for utilization of the six-step mode, as it has been discussed in [29]. Also, the presented implementation of the safeguard monitors only the momentary current, whereas a safeguarding of the average current (e.g., over the course of one electric rotation) is equally safe but allows to utilize the PMSM's capability much more effectively.

In terms of hardware effort, the proposed approach is quite costly, mainly due to the utilization of FPGA resources. Prospectively, this will not remain an issue because the calculation performance of embedded hardware increases, while its cost decreases with the progress of industrial development. An industrial application of the DQ-DTC is, therefore, not easily affordable in the short term, but further development and gain in knowledge in data-driven motor control is targeted building upon this contribution.

2) Further Scope: Apart from the DQ-DTC, the presented insights can also be applied to different drive systems. Most evidently, it would be of interest to transfer the DQ-DTC, which is an FCS setup, to the CCS, where modulators are used to set the average voltage. Although RL approaches for CCS environments are readily available, the safeguarding procedure would need a major overhaul for such scenarios. The proposed reward function could presumably be retained.

Further, the given functionality should also be conceivable for externally excited synchronous machines, which feature further degrees of freedom concerning their operating strategy, and nonsynchronous motors such as induction drives. For the latter, the full state vector describing the environment's state is not inherently measurable due to missing rotor flux linkage sensors, necessitating data-driven observation methods that could be implemented in an explicit or implicit fashion. Outside the domain of electric drives, also power grid applications as well as power electronic devices could be handled in an RL-based FCS scheme as presented in this article.

REFERENCES

[1] W. Peters, O. Wallscheid, and J. Böcker, "Optimum efficiency control of interior permanent magnet synchronous motors in drive trains of electric and hybrid vehicles," in *Proc. 17th Eur. Conf. Power Electron. Appl.*, 2015, pp. 1–10.

[2] R. Gabriel, W. Leonhard, and C. J. Nordby, "Field-oriented control of a standard AC motor using microprocessors," *IEEE Trans. Ind. Appl.*, vol. IA-16, no. 2, pp. 186–192, Mar. 1980.

[3] I. Takahashi and T. Noguchi, "A new quick-response and high-efficiency control strategy of an induction motor," *IEEE Trans. Ind. Appl.*, vol. IA-22, no. 5, pp. 820–827, Sep. 1986.

[4] P. Karamanakos, E. Liegmann, T. Geyer, and R. Kennel, "Model predictive control of power electronic systems: Methods, results, and challenges," *IEEE Open J. Ind. Appl.*, vol. 1, pp. 95–114, Aug. 2020.

[5] T. Schindler and A. Dietz, "Real-time inference of neural networks on FPGAs for motor control applications," in *Proc. Int. Electric Drives Prod. Conf.*, 2020, pp. 1–6.

[6] M. Schenke, W. Kirchgässner, and O. Wallscheid, "Controller design for electrical drives by deep reinforcement learning: A proof of concept," *IEEE Trans. Ind. Inform.*, vol. 16, no. 7, pp. 4650–4658, Jul. 2020.

[7] G. Book et al., "Transferring online reinforcement learning for electric motor control from simulation to real-world experiments," *IEEE Open J. Power Electron.*, vol. 2, pp. 187–201, Mar. 2021.

[8] P. Balakrishna, G. Book, W. Kirchgässner, M. Schenke, A. Traue, and O. Wallscheid, "Gym-electric-motor (GEM): A python toolbox for the simulation of electric drive systems," *J. Open Source Softw.*, vol. 6, no. 58, 2021, Art. no. 2498. [Online]. Available: https://doi.org/10.21105/joss.02498

[9] M. Schenke and O. Wallscheid, "A deep Q-learning direct torque controller for permanent magnet synchronous motors," *IEEE Open J. Ind. Electron. Soc.*, vol. 2, pp. 388–400, Apr. 2021.

[10] M. Schenke, B. Haucke-Korber, and O. Wallscheid, "Coffee machine vs. machine learning: Who is Quicker?" 2023. [Online]. Available: https://www.youtube.com/watch?v=hQ49Mc6LV78

[11] M. Schenke, B. Haucke-Korber, and O. Wallscheid, "EdgeRL pipeline," 2022. [Online]. Available: https://github.com/max-schenke/EdgeRL

[12] R. De Doncker, D. Pulle, and A. Veltman, *Advanced Electrical Drives: Analysis, Modeling, Control(Power Systems)*. Dordrecht, Netherlands: Springer, 2010.

[13] C. Hackl, J. Kullick, and N. Monzen, "Generic loss minimization for nonlinear synchronous machines by analytical computation of optimal reference currents considering copper and iron losses," in *Proc. IEEE Int. Conf. Ind. Technol.*, 2021, pp. 1348–1355.

[14] V. Mnih et al., "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[15] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–33, Feb. 2015.

[16] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[18] L. Brunke et al., "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 5, no. 1, pp. 411–444, 2022, doi: 10.1146/annurev-control-042920-020211.

[19] A. Brosch, S. Hanke, O. Wallscheid, and J. Böcker, “Data-driven recursive least squares estimation for model predictive current control of permanent magnet synchronous motors,” *IEEE Trans. Power Electron.*, vol. 36, no. 2, pp. 2179–2190, Feb. 2021.

[20] S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Discovering governing equations from data by sparse identification of nonlinear dynamical systems,” *Proc. Nat. Acad. Sci.*, vol. 113, no. 15, pp. 3932–3937, 2016. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1517384113

[21] C.-K. Lin, T.-H. Liu, J.-T. Yu, L.-C. Fu, and C.-F. Hsiao, “Model-free predictive current control for interior permanent-magnet synchronous motor drives based on current difference detection technique,” *IEEE Trans. Ind. Electron.*, vol. 61, no. 2, pp. 667–681, Feb. 2014.

[22] Y. Zhang, J. Jin, and L. Huang, “Model-free predictive current control of PMSM drives based on extended state observer using ultralocal model,” *IEEE Trans. Ind. Electron.*, vol. 68, no. 2, pp. 993–1003, Feb. 2021.

[23] J. Holtz, W. Lotzkat, and A. Khambadkone, “On continuous control of PWM inverters in the overmodulation range including the six-step mode,” *IEEE Trans. Power Electron.*, vol. 8, no. 4, pp. 546–553, Oct. 1993.

[24] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, “Safe reinforcement learning via shielding,” 2017. [Online]. Available: http://arxiv.org/abs/1708.08611

[25] K. P. Wabersich and M. N. Zeilinger, “Linear model predictive safety certification for learning-based control,” 2018. [Online]. Available: http://arxiv.org/abs/1803.08552

[26] J. Löfberg, “Oops! I. Cannot do it again: Testing for recursive feasibility in MPC,” *Automatica*, vol. 48, no. 3, pp. 550–555, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109811005723

[27] M. Agarwal and V. Aggarwal, “Blind decision making: Reinforcement learning with delayed observations,” in *Proc. Int. Conf. Automat. Plan. Scheduling*, 2021, vol. 31, pp. 2–6. [Online]. Available: https://ojs.aaai.org/index.php/ICAPS/article/view/15940

[28] A. Brosch, O. Wallscheid, and J. Böcker, “Long-term memory recursive least squares online identification of highly utilized permanent magnet synchronous motors for finite-control-set model predictive control,” *IEEE Trans. Power Electron.*, vol. 38, no. 2, pp. 1451–1467, Feb. 2023.

[29] B. Haucke-Korber, M. Schenke, and O. Wallscheid, “Reinforcement learning-based deep Q. direct torque control with adaptable switching frequency towards six-step operation of permanent magnet synchronous motors,” in *Proc. IKMT GMM/ETG Symp.*, 2022, pp. 1–6.

Maximilian Schenke received the bachelor’s and master’s (hons.) degrees in electrical engineering, in 2017 and 2019, respectively, from Paderborn University, Paderborn, Germany, where he is currently working toward the doctoral degree in electrical engineering with the Department of Power Electronics and Electrical Drives.

His research interests include the application of reinforcement learning methods in the domain of drive control.

Barnabas Haucke-Korber (Member, IEEE) received the bachelor’s and master’s degrees in electrical engineering from the Nuremberg Institute of Technology, Nuremberg, Germany, in 2017 and 2020, respectively.

He is currently working as a Research Associate with the Department of Power Electronics and Electrical Drives, Paderborn University, Paderborn, Germany. From 2020 to 2021, he worked as a Development Engineer with KARING GmbH, Priesendorf, Germany. His research interests include optimal control of electrical drives and efficient real-time implementation of control algorithms on control platforms.

Oliver Wallscheid (Senior Member, IEEE) received the bachelor’s and master’s (Hons.) degrees in industrial engineering and the doctorate (Hons.) degree in electrical engineering from the Paderborn University, Paderborn, Germany, in 2010, 2012, and 2017, respectively.

Since 2017, he has been a Senior Research Fellow with the Department of Power Electronics and Electrical Drives, Paderborn University. His research interests include data-driven identification and intelligent control of electrical power systems in decentralized grids, power electronics, and drives.