IEEE Open Journal of the
Industrial Electronics Society

Received 2 March 2021; revised 1 April 2021; accepted 17 April 2021. Date of publication 26 April 2021;
date of current version 7 June 2021. The review of this paper was arranged by Associate Editor Mingxi Liu.

Digital Object Identifier 10.1109/OJIES.2021.3075521

A Deep Q-Learning Direct Torque Controller for Permanent Magnet Synchronous Motors

MAXIMILIAN SCHENKE [1] AND OLIVER WALLSCHEID [2] (Member, IEEE)

[1]Department of Power Electronics and Electrical Drives, Paderborn University, 33098 Paderborn, Germany
[2]Department of Automatic Control, Paderborn University, 33098 Paderborn, Germany

CORRESPONDING AUTHOR: MAXIMILIAN SCHENKE (e-mail: schenke@lea.uni-paderborn.de)

This work was supported by computing time provided by the Paderborn Center for Parallel Computing (PC^2).

ABSTRACT Torque control of electric drives is a challenging task, as high dynamics need to be achieved despite different input and state constraints while also pursuing secondary objectives, e.g., maximizing power efficiency. Whereas most state-of-the-art methods generally necessitate thorough knowledge about the system model, a model-free deep reinforcement learning torque controller is proposed. In particular, the deep Q-learning algorithm is utilized which has been successfully used in different application scenarios with a finite action set in the recent past. This nicely fits the considered system, a permanent magnet synchronous motor supplied by a two-level voltage source inverter, since the latter is a power supply unit with a limited amount of distinct switching states. This contribution investigates the deep Q-learning finite control set framework and its design, including the conception of a reward function that incorporates the demands concerning torque tracking, efficiency maximization and compliance with operation limits. In addition, a comprehensive hyperparameter optimization is presented, which addresses the many degrees of freedom of the deep Q-learning algorithm striving for an optimal controller configuration. Advantages and remaining challenges of the proposed algorithm are disclosed through an extensive validation, which includes a direct comparison with a state-of-the-art model predictive direct torque controller.

INDEX TERMS Data-driven control design, deep Q-learning, direct torque, hyperparameter optimization, nonlinear control systems, permanent magnet motors, reinforcement learning.

I. INTRODUCTION

The utilization of sophisticated and highly tailored model-based controller designs has been a standard in electric drive control for the past fifty years. The available AC drive control methods range from linear proportional-integral (PI) controllers based on the concept of field-oriented control (FOC) [1] and direct torque control (DTC) [2] to model predictive control (MPC) [3]. Especially in the past twenty years, the concept of MPC has been investigated in depth, with a comprehensive set of improvements to the original current control concept [4]. Furthermore, the emerging consideration of combining approaches like model predictive direct torque control (MP-DTC) [5], [6] enables for state-of-the-art torque dynamics. A weakness that all these model-based control designs have in common is their dependence on plant-specific system knowledge. The control engineer needs to have a parameterized drive model readily available in order to develop a torque controller of satisfactory performance. Here, the accuracy of the plant knowledge is crucial for the resulting control quality. Accordingly, a significant fraction of the development time is spent with plant modeling and parameter identification tasks [7], [8], even before the actual controller design can begin. Performance losses due to unavoidable modeling inaccuracies still have to be accepted in the final outcome.

Another bottleneck is the required calculation time. As the linear PI controller dictates an explicit control policy, the necessary processing capacity for this control scheme on its own is rather low. If, however, online identification tasks need to be considered in order to adapt the controller to the demands of the drive system's momentary operating point, the processing time can increase dramatically. Since MPC oftentimes also relies on online parameter identification, the design process of such a controller usually incorporates the task to split the

turnaround time appropriately between the model identification and solving the optimal control problem.

The utilization of machine learning for the domain of drive control has recently made some advances, e.g., for the compensation of parasitic inverter behavior [9] or the estimation of drive temperature [10]. However, the potential of machine learning methods for drive controllers that could overcome the above-mentioned difficulties is yet to be explored. This article is dedicated to contribute to this endeavor.

A. STATE-OF-THE-ART

In optimal control scenarios, the control problem can be defined as a dynamic optimization task:

$$\begin{aligned} &\min_{\boldsymbol{u}} \sum_{i=k}^{\infty} \gamma^{i-k} r(\boldsymbol{x}_k, \boldsymbol{u}_k), \\ \text{s.t.} \quad &\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k), \\ &\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \boldsymbol{0}, \\ &\boldsymbol{l}(\boldsymbol{x}_k, \boldsymbol{u}_k) \leq \boldsymbol{0}. \end{aligned} \tag{1}$$

Herein, the discrete time index is denoted by k, the system state and system input are denoted by $\boldsymbol{x}$ and $\boldsymbol{u}$, respectively.[1] The discount factor $\gamma \in [0, 1[$ allows to shift the focus of the control task between short-sighted and far-sighted perspective. The cost r evaluates the control performance within each step. In addition to the optimization task, the system state is subject to the dynamics of the corresponding plant system $\boldsymbol{f}$, and the state and input variables need to comply with operation limitations $\boldsymbol{h}$ and $\boldsymbol{l}$.

For the application of power electronic-driven systems, two general controller classes can be formulated. In continuous-control-set (CCS) problems, the optimization problem in (1) has to be solved considering real-valued, continuous input voltages $\boldsymbol{u}$. This applies whenever the drive voltage source inverter receives its input signals from an intermediate modulator. In contrast, this contribution focuses finite-control-set (FCS) problems where the control output is limited to discrete actions. In the drive control context this corresponds to the distinct switching states of the power electronic inverter. Accordingly, FCS-MPC applications target the selection of optimal switching commands to solve the optimization problem in (1).

Since exhaustive search to find optimal switching sequences is unfeasible for the infinite time horizon specified in (1), FCS-MPC is only applicable if the optimal control problem (1) is considered on a finite time horizon $N << \infty$ (hence, allowing $\gamma = 1$) [11]. Simplification methods may then be employed to facilitate the optimization, which can permit a slight increase of N, e.g., utilizing sphere decoding [12], [13] or by using heuristic search algorithms [14].

An interesting, emerging alternative to solving (1) is reinforcement learning (RL). Here, the control design is shifted to a data-driven approach [15]–[17]. While MPC is mostly based on online optimization, the RL agent optimizes its control policy during the training phase, resulting in an explicit and, therefore, high-performance controller after the training is finished. This, however, only holds given the assumption that the training phase sufficiently covers the operation range which is relevant for the later usage. During training, the RL controller (usually denoted as the agent) learns about the drive characteristics by direct interaction with the system. The RL controller is therefore a model-free approach since it does not require any model information beforehand.

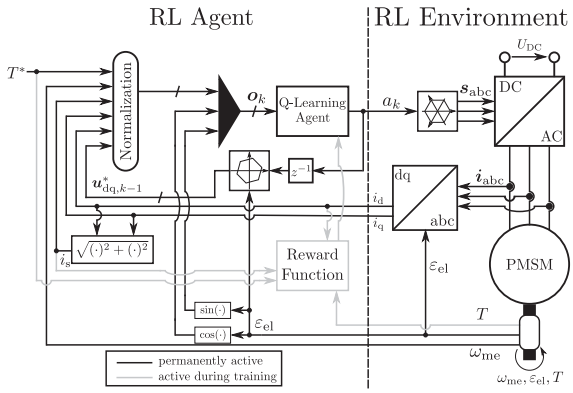

FIG. 1. Schematic of the overall control structure.

In RL, the optimization problem in (1) is usually reformulated as a maximization problem, rendering r a reward signal. It is inherently viewed for an infinite time horizon without the problem of growing computational effort. In order to avoid convergence problems in this case, the discount factor then needs to be lower than one ($\gamma \in [0, 1[$).

While first promising CCS-RL approaches for the current control problem of drives are already available [18]–[20], additional control tasks, such as torque or speed control, and learning FCS approaches have not yet been investigated. The question therefore arises if learning, model-free controllers can also be successfully applied to further, challenging control tasks in drive and power electronic applications.

B. CONTRIBUTION

In the following investigation, FCS torque control of a permanent magnet synchronous motor (PMSM) fed by a two-level inverter is addressed. In particular, this contribution utilizes a deep Q-learning agent [21] to solve the described control problem. This kind of RL concept is customized to deal with continuous state measurements and discrete control inputs (actions). Therefore, the resulting control approach is labeled deep Q direct torque control (DQ-DTC). A schematic of the overall controller structure is presented in Fig. 1. Compared to established, model-based approaches, the DQ-DTC method offers the following potentials:

- The same preconfigured learning algorithm can be applied to any new PMSM drive system.

[1] Bold symbols denote multidimensional quantities (e.g., matrices and vectors).

- Plant-specific expert knowledge is not required (e.g., motor parameters do not need to be available).
- Parasitic effects such as (cross-)saturation, iron losses and temperature influences do not need to be modeled or identified since they are included within the learned control policy.
- The entire constant torque and flux weakening range are covered by an explicit torque control algorithm.

In order to deliver a comprehensive investigation of the DQ-DTC approach, this contribution will firstly present the theoretical fundamentals of electrical drives and the deep Q-learning concept, followed by a review of the available domain-specific knowledge that can be utilized to enable expedient and generalizable training. The learning framework is introduced focusing on the coverage of the state space using exploring starts. A hyperparameter optimization process is performed in order to find an optimally configured agent for a PMSM drive environment. Lastly, the DQ-DTC is compared to the related MP-DTC approach from the MPC domain.

II. DRIVE SYSTEM MODEL

The drive system under investigation consists of a B6-bridge power electronic voltage source inverter that supplies a PMSM.

A. PERMANENT MAGNET SYNCHRONOUS MOTOR

The three-phase PMSM can be modeled concisely when using rotor-fixed dq-coordinates. In order to transform the three-phase voltages $\boldsymbol{u}_{\mathrm{abc}}$ and currents $\boldsymbol{i}_{\mathrm{abc}}$ to the two-dimensional, rotor-fixed representation, the following transformation is employed:

$$\begin{bmatrix} u_{\mathrm{d}} \\ u_{\mathrm{q}} \end{bmatrix} = \begin{bmatrix} \cos(\varepsilon_{\mathrm{el}}) & \sin(\varepsilon_{\mathrm{el}}) \\ -\sin(\varepsilon_{\mathrm{el}}) & \cos(\varepsilon_{\mathrm{el}}) \end{bmatrix} \underbrace{\begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} u_{\mathrm{a}} \\ u_{\mathrm{b}} \\ u_{\mathrm{c}} \end{bmatrix}}_{=\begin{bmatrix} u_\alpha & u_\beta \end{bmatrix}^{\mathrm{T}}}, \tag{2}$$

wherein the electrical rotor angle is denoted by $\varepsilon_{\mathrm{el}}$. The intermediate result is given by the two-dimensional, stator-fixed voltage $\boldsymbol{u}_{\alpha\beta}$. Using the dq representation, the PMSM can be characterized by the fundamental wave model [22]:

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}_{\mathrm{dq}}(t) + \boldsymbol{e}(t), \\ &\boldsymbol{x}(t) = \begin{bmatrix} i_{\mathrm{d}}(t) \\ i_{\mathrm{q}}(t) \end{bmatrix}, \boldsymbol{u}_{\mathrm{dq}}(t) = \begin{bmatrix} u_{\mathrm{d}}(t) \\ u_{\mathrm{q}}(t) \end{bmatrix}, \boldsymbol{e}(t) = \begin{bmatrix} 0 \\ -\frac{p\omega_{\mathrm{me}}(t)\psi_{\mathrm{p}}}{L_{\mathrm{q}}} \end{bmatrix}, \\ &\boldsymbol{A}(t) = \begin{bmatrix} -\frac{R_{\mathrm{s}}}{L_{\mathrm{d}}} & \frac{p\omega_{\mathrm{me}}(t)L_{\mathrm{q}}}{L_{\mathrm{d}}} \\ -\frac{p\omega_{\mathrm{me}}(t)L_{\mathrm{d}}}{L_{\mathrm{q}}} & -\frac{R_{\mathrm{s}}}{L_{\mathrm{q}}} \end{bmatrix}, \boldsymbol{B}(t) = \begin{bmatrix} \frac{1}{L_{\mathrm{d}}} & 0 \\ 0 & \frac{1}{L_{\mathrm{q}}} \end{bmatrix}, \\ &i_{\mathrm{s}}(t) = \sqrt{\left(i_{\mathrm{d}}(t)\right)^2 + \left(i_{\mathrm{q}}(t)\right)^2}, \\ &T(t) = \frac{3}{2}p\left(\psi_{\mathrm{p}} + \left(L_{\mathrm{d}} - L_{\mathrm{q}}\right)i_{\mathrm{d}}(t)\right)i_{\mathrm{q}}(t), \end{aligned} \tag{3}$$

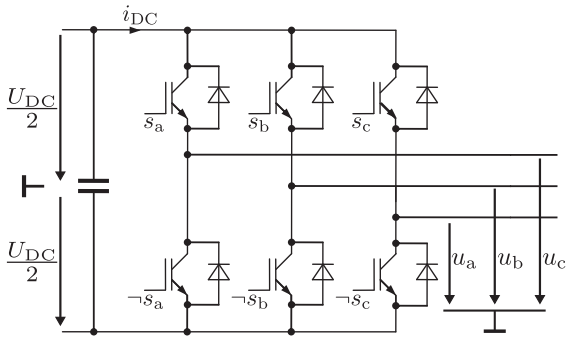

FIG. 2. Circuit diagram of the three-phase B6-bridge voltage source inverter.

TABLE I Correspondence Between the Control Action, the Switching Commands and the 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

with the d and q inductances L_{d}, L_{q}, the stator resistance R_{s}, the permanent magnetic flux ψ_{p}, the pole-pair number p, the mechanical angular velocity ω_{me}, the stator current i_{s} and the generated drive torque T. This model will be utilized for the simulative analysis, which is conducted based on the open-source control simulation software gym-electric-motor (GEM) [23].

B. B6-BRIDGE POWER CONVERTER

The B6-bridge inverter is depicted in Fig. 2. In the given context of FCS control, the action space of this inverter consists of eight distinguishable switching states (which already excludes all switching states that result in half-bridge short circuits). The task of any FCS-type controller is to choose the switching state which is most preferable to optimize a given reward or cost function. From a control and learning point of view, one major challenge is to handle nonlinear mapping of switching commands a, switching states $\boldsymbol{s}_{\mathrm{abc}}$ and three-phase voltages $\boldsymbol{u}_{\mathrm{abc}}$ as shown in Table I. In the scope of this article, the DC-link voltage U_{DC} is assumed to be constant.

C. DISCRETE-TIME CONTROL

In addition to the physical system behavior, the impact of time discretization for computerized control has to be considered. As depicted in Fig. 3, the action a_k is calculated based on the feature vector $\boldsymbol{o}_{k-1}$ (lifted measurement vector, will be discussed in Sec. III), but will be applied at time step k due to the digital control delay. The change of the system state, triggered by a_k, will become visible in $\boldsymbol{o}_{k+1}$. This one-step computational delay between the application of an action and

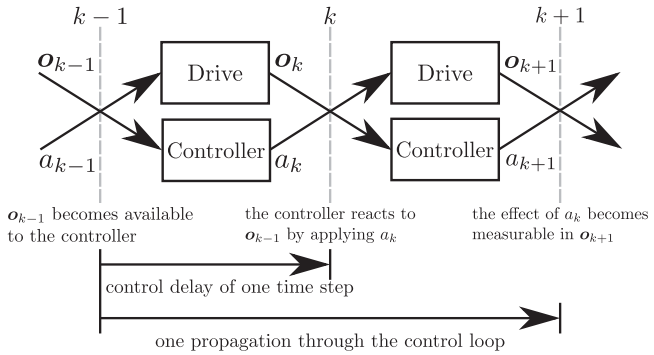

FIG. 3. Chronology of the digital control loop.

the measurability of its effect poses another challenge when designing the control agent. As the drive system behaves deterministically, the prediction of $\boldsymbol{o}_k$ is possible if $\boldsymbol{o}_{k-1}$ and a_{k-1} are known. In the MPC domain this control delay compensation is easily realized via another model-based prediction step. Model-free RL approaches must learn to implicitly compensate the influence of such a dead time by means of their data-driven training process before satisfactory control quality can be achieved. This knowledge will later be integrated into the agent's design.

III. DEEP Q-LEARNING

The deep Q-learning algorithm is one of the most established RL algorithms for scenarios with a real-valued feature vector $\boldsymbol{o}$, which is derived from the system's measurement, and a discrete action a [21]. Its goal is to maximize the return g, which is the cumulative, discounted reward over time:

$$g_k = \mathbb{E}\left\{\sum_{i=k}^{\infty} \gamma^{i-k} R_{i+1} \middle| \boldsymbol{O}_k = \boldsymbol{o}_k, A_k = a_k\right\}. \tag{4}$$

Herein, capital letters denote random variables and $\mathbb{E}\{\cdot\}$ denotes the expected value. The infinite horizon series presented in (4) can be reformulated to resemble a dynamic one-step problem by the Bellman equation [24]:

$$\begin{aligned} g_k &= q(\boldsymbol{o}_k, a_k) \\ &= \mathbb{E}\{R_{k+1} + \gamma \underbrace{q(\boldsymbol{O}_{k+1}, A_{k+1})}_{g_{k+1}} | \boldsymbol{O}_k = \boldsymbol{o}_k, A_k = a_k\}. \end{aligned} \tag{5}$$

This equation recursively defines the action value q, which is computed by means of the immediate reward r_{k+1} and by the discounted future action value. Thus, the action value represents the expectable return which is based on the momentary state-action pair. For a given action-value function, the optimal action a^* on the FCS can be determined by

$$a_k^* = \arg\max_{a'} q(\boldsymbol{o}_k, a'). \tag{6}$$

Therefore, one major challenge in Q-learning is to determine an action-value approximation that allows an accurate mapping $q : (\boldsymbol{o}, a) \to g$ in an online fashion, which is challenging if the system dynamics or the reward behavior are nonlinear and can vary at runtime. The desired mapping can be found by combining the flexibility of deep neural networks with the Q-learning algorithm, leading to an action-value approximator network $\hat{q}_{\boldsymbol{\theta}}$ with $\boldsymbol{\theta}$ representing the network weights[2]. This representation of the action value function is popularly known as deep Q-network (DQN), which is the core of the deep Q-learning algorithm. In order to train such a DQN, the Bellman equation in (5) can be rearranged to yield a mean squared error minimization problem [25]

$$\begin{aligned} &\min_{\boldsymbol{\theta}} J_Q \\ &\text{s.t. } J_Q = \frac{1}{|\mathcal{B}|} \sum_{\mathcal{E}_k \in \mathcal{B}} (\hat{q}_{\boldsymbol{\theta}}(\boldsymbol{o}_k, a_k) - \\ &(r_{k+1} + \gamma(1 - d_{k+1}) \max_{a'} \hat{q}_{\boldsymbol{\theta}_{\text{target}}}(\boldsymbol{o}_{k+1}, a')))^2, \end{aligned} \tag{7}$$

wherein a done flag $d = 1$ would mark the termination of the control episode (e.g., due to violations of the operation limits), consequently nullifying the look-ahead action value. An optimization step can be performed whenever a state transition experience $\mathcal{E}_k = \{\boldsymbol{o}_k, a_k, r_{k+1}, \boldsymbol{o}_{k+1}, d_{k+1}\}$ is available, but usually, the transitions are processed in mini batches $\mathcal{B}$, leading to more robust improvements of the DQN.

When investigating (7) a little further, it can be seen that the DQN comes up within the estimator as well as in the estimation target, allowing efficient updates at runtime without the need for another estimation mechanism. The practice of updating an estimation towards an estimation is commonly known as bootstrapping [16]. Within bootstrapping methods, there is a risk of rising parameter oscillations if the estimation target is too volatile. Therefore, the best practice of using a target network $\hat{q}_{\boldsymbol{\theta}_{\text{target}}}$ has been established for such bootstrapping methods [21]. The target network contains a set of target parameters $\boldsymbol{\theta}_{\text{target}}$ which are derived from the original parameters $\boldsymbol{\theta}$ in order to dampen emerging parameter oscillations. This can be achieved either by low-pass filtering the original parameters

$$\boldsymbol{\theta}_{\text{target}} \leftarrow (1 - \rho)\boldsymbol{\theta}_{\text{target}} + \rho\boldsymbol{\theta}, \quad \rho \in]0, 1], \tag{8}$$

or by updating the target parameters $\boldsymbol{\theta}_{\text{target}}$ only occasionally:

$$\boldsymbol{\theta}_{\text{target}} \leftarrow \boldsymbol{\theta}, \quad \text{after every } \rho \in \mathbb{N} \text{ steps}. \tag{9}$$

The target update parameter ρ can be considered a hyperparameter of the training procedure. Its optimal choice is problem-dependent and will be investigated during hyperparameter optimization in Section V-B.

Another trait of the Q-learning algorithm is the assumption of the optimal look-ahead action value, as shown in (7). This characteristic implements a maximization bias, as it only yields realistic estimations if the system behavior is deterministic and, thus, reproducible. Since this can usually be assumed for technical applications like electric drive control, no further measures are necessary here. However, attention is

[2] $\hat{x}$ denotes an estimated quantity.

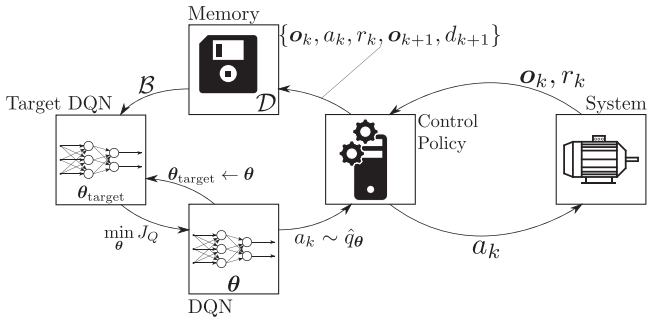

FIG. 4. **Schematic depiction of the DQN learning routine.**

needed when transferring this concept to stochastic systems or systems with very dominant measurement noise. For the latter cases, the usage of double Q-learning should be considered instead [26].

Lastly, the DQN agent performs exploration, i.e., evaluating new actions which deviate from the current control policy. This step is required to find better policies and is implemented using the so-called ϵ-greedy policy:[3]

$$a_k = \begin{cases} \arg\max_{a'} q(\boldsymbol{o}_k, a'), & \text{with probability } 1-\epsilon. \\ \text{a random element from } \mathcal{A}, & \text{with probability } \epsilon. \end{cases} \tag{10}$$

With the set of available actions $\mathcal{A}$. After finishing the training phase, the ϵ parameter is usually set to zero, in order to yield reproducible performance. Especially in the context of technical process control, randomness is unwanted after finishing the training phase, resulting in $\epsilon = 0$ to enable reliable controller validation and usage. A schematic of the DQN learning routine is depicted in Fig. 4, the pseudocode of the presented deep Q-learning algorithm is given in Alg. 1.

Algorithm 1: Deep Q-Learning Pseudocode.

randomly initialize weights $\boldsymbol{\theta}$ of $\hat{q}_{\boldsymbol{\theta}}(\boldsymbol{o}, a)$
initialize target weights accordingly $\boldsymbol{\theta}_{\text{target}} \leftarrow \boldsymbol{\theta}$
initialize replay memory $\mathcal{D}$ with capacity $|\mathcal{D}|$
$m \leftarrow 0$
while $m < M$ **do**
  $k \leftarrow 0$
  obtain $\boldsymbol{o}_0$ and d_0
  while $k < K$ **do**
    select a random action $a_k \in \mathcal{A}$ with probability ϵ
    otherwise select $a_k = \arg\max_{a'} \hat{q}_{\boldsymbol{\theta}}(\boldsymbol{o}_k, a')$
    execute a_k and obtain $r_{k+1}, \boldsymbol{o}_{k+1}, d_{k+1}$
    save transition experience to replay buffer
      $\mathcal{D} \leftarrow \{\boldsymbol{o}_k, a_k, r_{k+1}, \boldsymbol{o}_{k+1}, d_{k+1}\}$
    sample experience minibatch $\mathcal{B} \subset \mathcal{D}$ of the size $|\mathcal{B}|$
    update $\boldsymbol{\theta}$ by minimizing J_Q on $\mathcal{B}$, (7)
    if $\rho \in]0, 1]$ **then**
      update target weights according to
        $\boldsymbol{\theta}_{\text{target}} \leftarrow (1-\rho)\boldsymbol{\theta}_{\text{target}} + \rho\boldsymbol{\theta}$
    else if $\rho \in \mathbb{N}$ **then**
      update target weights according to
        $\boldsymbol{\theta}_{\text{target}} \leftarrow \boldsymbol{\theta}$ whenever $m \bmod \rho = 0$
    end if
    if $d_{k+1} = 1$ **then**
      reset the environment and **break**
    end if
    $k \leftarrow k + 1$
    $m \leftarrow m + 1$
  end while
end while

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

The considered control problem of torque control necessitates a control agent that chooses the optimal inverter switching state such that the primary condition of adherence to the reference torque T^* has priority over the secondary condition of maximum efficiency, which translates to minimal stator current i_s while still maintaining the reference torque [27]. These demands can be formulated as

$$\begin{aligned} &\min_{a_k} i_{\text{s},k} \\ \text{s.t.} \quad &T_k = T_k^*, \\ &i_{\text{s},k} \leq i_\text{lim}, \end{aligned} \tag{11}$$

with the current limit i_lim. Although the motor current and torque behavior highly depend on the individual drive system, the DQ-DTC agent does not have any access to a plant model and, therefore, must learn a policy to optimize (11) using Alg. 1 purely in a data-driven fashion.

In order to achieve satisfactory controller performance with the DQ-DTC approach, it is therefore sensible to design the state feature vector $\boldsymbol{o}$ and the reward function r with respect to the domain-specific knowledge about the PMSM. Since plant-specific knowledge will not be integrated into the design process, this approach is directly transferable to the large class of synchronous motors (SMs), e.g., the surface permanent magnet SM (where $L_\text{d} = L_\text{q}$), the synchronous reluctance motor (where $\psi_\text{p} = 0$) and the highly utilized PMSM (where $L_\text{d} = L_\text{d}(\boldsymbol{i}_\text{dq})$ and $L_\text{q} = L_\text{q}(\boldsymbol{i}_\text{dq})$).

Domain-specific expert knowledge (which is available without comprehensive plant analysis) can be implemented by appropriately designing the feature vector $\boldsymbol{o}$, which can be considered as feature engineering. As will be discussed later in detail, the feature vector is based only on the available measurement signals during drive operation but lifted into a higher feature space enriching the learning process. Another exploitable design decision has to be made when constructing the reward function r such that the optimization problem (11) can be solved sufficiently and with respect to the operation

[3]The greek letter epsilon is commonly used within the drive domain („rotor angle ε_el“), as well as in the RL domain („ϵ-greedy policy“) In this contribution the distinguishable symbols ε (within the electric drive domain) and ϵ (within the RL domain) were used to avoid breaking with the naming conventions of both communities.

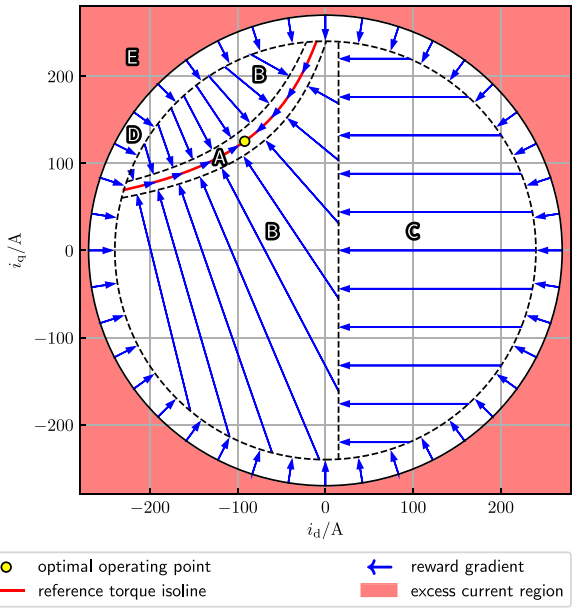

FIG. 5. Schematic of the operation regions, reward gradients and optimal operating point at low speed.

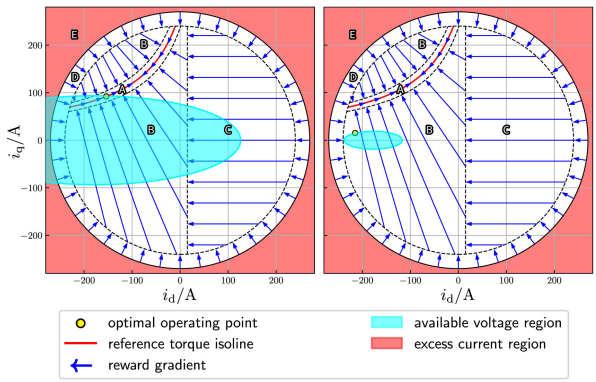

FIG. 6. Shrinking of the reachable operation range as implied by (12) at moderate speed (left) and high speed (right).

limitations of the PMSM. Naturally, these degrees of freedom must be configured without knowledge about the individual drive parameters.

Since the formal definition of the control problem (11) is not readily suitable for DQN learning, a proper reward function design reproducing these control objectives is to be discussed in the following.

A. OPERATION LIMITATIONS

As any technical process the operation range of the drive system underlies physical boundaries whose violation has to be avoided for safety reasons, and reasonable operation conditions whose violation reduces the efficiency and should therefore be avoided whenever possible. A depiction of the corresponding operation regions is given in Fig. 5. In the case of the PMSM, the primary operation limit is given by the current limit i_{lim}, which should not be surpassed in order to prevent the motor and inverter from overheating (region $\mathbb{E}$). The nominal current $i_{\mathrm{n}} < i_{\mathrm{lim}}$ is the maximum current the motor can endure permanently. It can be exceeded for short time periods, but a long-term operation in respective states is not advisable (region $\mathbb{D}$). A secondary operation condition is motivated by the current i_{d}, which corresponds to the magnetic field within the PMSM. For efficiency reasons, operating at positive i_{d} values must be prevented (region $\mathbb{C}$). However, the corresponding boundary $i_{\mathrm{d}+}$ should be allowed to be slightly greater than zero, since low-torque operation points, which naturally have low current demands, will be maintained more precisely if the current ripple has a little leeway. As already stated in Sec. IV, the general objective of the DQ-DTC is to track the reference torque T^* (region $\mathbb{B}$) while minimizing stator current (region $\mathbb{A}$). Naturally, these performance criteria have a lower priority than the aforementioned safety criteria. The significance of each named condition is ordered hierarchically, from $\mathbb{A}$ being least important, to $\mathbb{E}$ being most important, which will be incorporated into the reward design.

Lastly, the DC-link voltage U_{DC} is also a limiting factor for the reachable torque references. At high velocity, the controller will use a large portion of the available voltage only to compensate for the generative induced voltage:

$$\left(L_{\mathrm{d}} i_{\mathrm{d}} + \psi_{\mathrm{p}}\right)^2 + \left(L_{\mathrm{q}} i_{\mathrm{q}}\right)^2 \leq \left(\frac{U_{\mathrm{DC}}}{\sqrt{3} p |\omega_{\mathrm{me}}|}\right)^2 . \tag{12}$$

This so-called flux weakening mode reduces the feasible current state space, potentially disabling the control system from reaching the optimal operation point as depicted in Fig. 6. Hence, the controller must target a different operation point in dependence of the motor speed: motor operation at low speed is to be optimized for efficiency, leading to maximum torque per current (MTPC) operation. At high speed, when operating at the voltage limit, higher torque output requires maximum torque per voltage (MTPV) characteristic [28]. The feature vector and the reward function must convey these demands without specific plant model knowledge.

B. FEATURE VECTOR DESIGN

The feature vector $\boldsymbol{o}$ at every time step k obtains the information about the momentary control system state. Thus, it needs to be designed such that it contains as much relevant information as possible without being bloated with redundant states. When considering the equation of torque generation (3), it is obvious that the motor currents $\boldsymbol{i}_{\mathrm{dq}}$ and the reference torque T^* should be implemented into $\boldsymbol{o}$. In usual drive applications the drive torque T is not measurable when considering economic factors. Within the scope of this contribution it is assumed that such a measurement is only available within the training phase (which could be conducted on a laboratory test bench setup where a torque sensor is usually in use), allowing for T to be considered for the reward function, but not for the feature vector. Thus, the control agent is forced to implicitly learn the mapping from current to torque, which includes the implicit learning of the corresponding parameters. In order to allow compensation of the computational delay mentioned in

Sec. II-C, the previously applied voltage $\boldsymbol{u}_{\mathrm{dq},k-1}$ is considered within the feature vector, enabling the DQN to implicitly predict $\boldsymbol{o}_{k+1}$, which is important for solving the Bellman equation in (5). The operation boundaries mentioned in Sec. IV-A further necessitate the consideration of the motor speed ω_{me} and the stator current amplitude i_{s}. Finally, the mapping of the static three-phase frame to the time-variant dq-frame depends on the motor angle $\varepsilon_{\mathrm{el}}$, which needs to be considered as well.

As usual in the field of machine learning, it is sensible to normalize the inputs and outputs of the DQN to the range of $[-1, 1]$ [29]. For most of the input variables this is easily done by dividing them by their corresponding limit value. For the motor angle $\varepsilon_{\mathrm{el}}$ it is more reasonable to make use of the $\cos(\cdot)$ and $\sin(\cdot)$ in order to provide a continuous and bounded angle information

$$\boldsymbol{o}_k = \Bigg[\frac{\omega_{\mathrm{me},k}}{\omega_{\mathrm{me,lim}}} \quad \frac{i_{\mathrm{d},k}}{i_{\mathrm{lim}}} \quad \frac{i_{\mathrm{q},k}}{i_{\mathrm{lim}}} \quad \frac{u_{\mathrm{d},k-1}}{\frac{U_{\mathrm{DC}}}{2}} \quad \frac{u_{\mathrm{q},k-1}}{\frac{U_{\mathrm{DC}}}{2}} \\ \kappa \cos(\varepsilon_{\mathrm{el},k}) \quad \kappa \sin(\varepsilon_{\mathrm{el},k}) \quad 2\frac{i_{\mathrm{s},k}}{i_{\mathrm{lim}}} - 1 \quad \frac{T_k^*}{T_{\mathrm{lim}}} \Bigg] \quad , \tag{13}$$

with the maximum mechanical angular velocity $\omega_{\mathrm{me,lim}}$ and the maximum reference torque T_{lim}. The strictly positive stator current i_{s} also needs to be offset in order to cover the range of $[-1, 1]$. The scaling parameter κ changes the value range of the $\cos(\cdot)$ and $\sin(\cdot)$ from $[-1, 1]$ to $[-\kappa, \kappa]$, which can be used to dampen the impact of the angle information if dominant harmonics are visible within the DQN output. A value of $\kappa = 0.1$ yielded good results in practice.

C. REWARD DESIGN

The reward function r has to be designed such that the optimization problem denoted in (11) is solved optimally under any given circumstance. This means that the highest achievable reward always has to correspond to the optimal operating point. Since the control agent seeks to maximize the reward through interaction, the reward function needs to be constructed with caution. As a first step, the range of estimated action values will be fixed to $q \in [-1, 1]$, resulting in the utilized DQN to have normalized outputs, which is beneficial for the training [29]. In case of system termination, no look-ahead action value will be considered, making it fairly easy to determine the termination reward:

$$q_{\mathrm{term}} = r_{\mathrm{term}} \overset{!}{=} -1. \tag{14}$$

This can be interpreted as a penalty whenever the DQN leaves the allowed state space, i.e., the RL agent is learning to stay within the system's limitations during the training phase.

For ongoing operation, the normalization can be achieved when considering the geometric series for the best and the worst case operation point:

$$q_{\max} = \sum_{i=k}^{\infty} \gamma^{i-k} r_{\max} = \frac{r_{\max}}{1-\gamma} \overset{!}{=} 1$$
$$\Leftrightarrow \quad r_{\max} = 1 - \gamma, \tag{15}$$

$$q_{\min} = \sum_{i=k}^{\infty} \gamma^{i-k} r_{\min} = \frac{r_{\min}}{1-\gamma} \overset{!}{=} -1$$
$$\Leftrightarrow \quad r_{\min} = -(1-\gamma). \tag{16}$$

With the computed reward range it is now possible to design the reward function in dependence of the active operation region as presented in Sec. IV-A. The resulting reward gradients, that will ultimately determine the targeted operation point, are depicted in Fig. 5 and Fig. 6.

$\mathbb{E}$ Excess current region
Entering this region will trigger an emergency system shutdown which in the RL sense corresponds to the termination of the episode. The learning is temporarily discontinued and the drive system must be re-initialized (starting a new episode). Minimum reward to discourage system termination is defined:

$$\text{if} \quad i_{\mathrm{lim}} < i_{\mathrm{s},k}: \\ r_k = -1, \qquad d_k = 1. \tag{17a}$$

$\mathbb{D}$ Short-time overcurrent region
Reward rises with decreasing stator current for safety reasons, i.e., the agent is encouraged to not permanently overload the system.

$$\text{if} \quad i_{\mathrm{n}} < i_{\mathrm{s},k} < i_{\mathrm{lim}}: \\ r_k = \left(1 - \frac{i_{\mathrm{s},k} - i_{\mathrm{n}}}{i_{\mathrm{lim}} - i_{\mathrm{n}}}\right) \frac{1-\gamma}{2} - (1-\gamma), \\ \Rightarrow r_k \in \left[-(1-\gamma), -\frac{1-\gamma}{2}\right], \qquad d_k = 0. \tag{17b}$$

$\mathbb{C}$ Unfavorable efficiency region
Although stable and safe operation can be achieved for positive i_{d} current, the resulting efficiency is inferior to operating the drive in the left i_{d}-i_{q}-half plane.

$$\text{if} \quad i_{\mathrm{s},k} < i_{\mathrm{n}} \quad \text{and} \quad i_{\mathrm{d+}} < i_{\mathrm{d},k}: \\ r_k = \left(1 - \frac{i_{\mathrm{d},k} - i_{\mathrm{d+}}}{i_{\mathrm{n}} - i_{\mathrm{d+}}}\right) \frac{1-\gamma}{2} - \frac{1-\gamma}{2}, \\ \Rightarrow r_k \in \left[-\frac{1-\gamma}{2}, 0\right], \qquad d_k = 0. \tag{17c}$$

$\mathbb{B}$ Desired operating region
Reward rises with decreasing torque error for tracking performance reasons:

$$\text{if } i_{\mathrm{s},k} < i_{\mathrm{n}} \text{ and } i_{\mathrm{d},k} < i_{\mathrm{d+}} \text{ and } T_{\mathrm{tol}} < \left|T_k^* - T_k\right|: \\ r_k = \left(1 - \left|\frac{T_k^* - T_k}{2T_{\mathrm{lim}}}\right|\right) \frac{1-\gamma}{2}, \\ \Rightarrow r_k \in \left[0, \frac{1-\gamma}{2}\right], \qquad d_k = 0. \tag{17d}$$

TABLE II Parameterization of the Considered Drive System

symbol	description	value	
p	pole-pair number	3	
R_s	stator resistance	17.932	mΩ
L_d	d inductance	0.37	mH
L_q	q inductance	1.2	mH
ψ_p	permanent magnet flux	65.65	mVs
i_n	nominal current	240	A
$i_\mathrm{d+}$	tolerable positive d current	15	A
i_lim	maximum current	270	A
U_DC	DC-link voltage	350	V
$\omega_\mathrm{me,lim}$	maximum angular velocity	1256.64	$\frac{1}{\mathrm{s}}$
T_n	nominal torque	150	N · m
T_lim	maximum reference torque	200	N · m
T_tol	torque control tolerance	5	N · m
$\mathrm{f_s}$	sampling frequency	20	kHz

𝔸 Reference torque isoline
Reward rises with decreasing stator current for efficiency reasons:

$$\text{if } i_{\mathrm{s},k} < i_\mathrm{n} \text{ and } i_{\mathrm{d},k} < i_\mathrm{d+} \text{ and } \left|T_k^* - T_k\right| < T_\mathrm{tol}:$$
$$r_k = \left(1 - \frac{i_{\mathrm{s},k}}{i_\mathrm{lim}}\right)\frac{1-\gamma}{2} + \frac{1-\gamma}{2},$$
$$\Rightarrow r_k \in \left[\frac{1-\gamma}{2}, 1-\gamma\right], \qquad d_k = 0. \tag{17e}$$

The reward functions (17a)-(17e) cover the entire motor operation region including constant torque and flux weakening operation. They represent the control objective from (11) without requiring knowledge of the specific drive behavior. Only the nominal and limit data of the drive should be (roughly) known in order to be able to perform the presented normalization of the reward functions.

V. SIMULATION RESULTS

To yield performance results, the PMSM drive system is simulated using the system model as presented in Sec. II, with further parameters given in Table II. The suggested DQ-DTC approach can now be optimized in terms of the DQN degrees of freedom (hyperparameters). Afterwards, the resulting control agent will be tested against a state-of-the-art MP-DTC implementation.

A. BASIC SETUP

The training and testing programs are implemented in Python 3.7. The drive system is simulated using the GEM library [23], the setup and training of the DQN is handled with the use of kerasRL [30], kerasRL2 [31] and tensorflow 2 [32]. The DQN weights are optimized using the Adam optimizer [33]. To force state-space exploration (i.e., to ensure state-space coverage), the training makes use of uniformly distributed exploring starts [16] on the controllable sub-space:

$$\omega_{\mathrm{me},0} \in [-\omega_\mathrm{me,lim}, \omega_\mathrm{me,lim}], \quad \varepsilon_{\mathrm{el},0} \in [-\pi, \pi],$$
$$i_{\mathrm{d},0} \in [\max(-i_\mathrm{n}, i_\mathrm{d,c,min}), \min(i_\mathrm{n}, i_\mathrm{d,c,max})],$$
$$i_{\mathrm{q},0} \in \left[\max\left(-\sqrt{i_\mathrm{n}^2 - i_{\mathrm{d},0}^2}, i_\mathrm{q,c,min}\right), \min\left(\sqrt{i_\mathrm{n}^2 - i_{\mathrm{d},0}^2}, i_\mathrm{q,c,max}\right)\right],$$
$$T_0^* \in [-T_\mathrm{lim}, T_\mathrm{lim}], \tag{18}$$

with

$$i_\mathrm{d,c,max} = -\frac{\psi_\mathrm{p}}{L_\mathrm{d}} + \frac{U_\mathrm{DC}}{L_\mathrm{d}\sqrt{3}p|\omega_{\mathrm{me},0}|},$$
$$i_\mathrm{d,c,min} = -\frac{\psi_\mathrm{p}}{L_\mathrm{d}} - \frac{U_\mathrm{DC}}{L_\mathrm{d}\sqrt{3}p|\omega_{\mathrm{me},0}|},$$
$$i_\mathrm{q,c,max} = \sqrt{\left(\frac{U_\mathrm{DC}}{L_\mathrm{q}\sqrt{3}p|\omega_{\mathrm{me},0}|}\right)^2 - \frac{L_\mathrm{d}^2}{L_\mathrm{q}^2}\left(i_{\mathrm{d},0} + \frac{\psi_\mathrm{p}}{L_\mathrm{d}}\right)^2}, \tag{19}$$
$$i_\mathrm{q,c,min} = -i_\mathrm{q,c,max}.$$

Herein, the maximum/minimum controllable i_d and i_q for the initialization are denoted by $i_\mathrm{d,c,max}, i_\mathrm{d,c,min}, i_\mathrm{q,c,max}$ and $i_\mathrm{q,c,min}$ respectively. The exploring starts are important to cover the entire drive's operation range during training. The model parameters used in (19) are unknown to the DQ-DTC agent and were only used to narrow down the training to the relevant working areas of the operation range.

Representative episodes of an exemplary training process are depicted in Fig. 7, Fig. 8 and Fig. 9. The training process starts out with a randomly initialized DQN, leading to

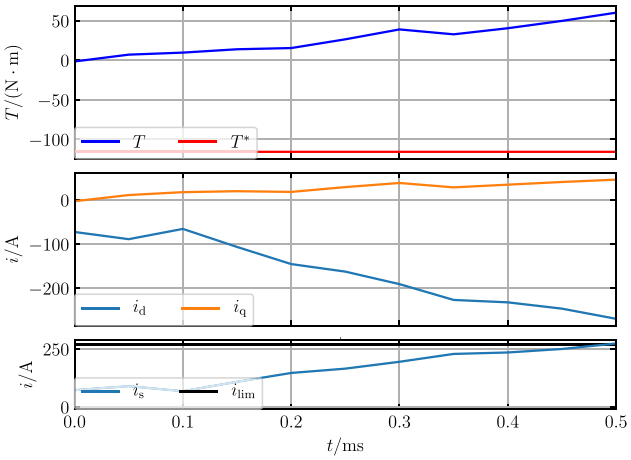

FIG. 7. Exemplary DQ-DTC performance in an early training episode, $\omega_\mathrm{me} = -1177,779\,\mathrm{s}^{-1}$.

TABLE III Evaluation of the Training Duration for the Learning Processes Depicted in Fig. 11 and Specification of the Workstation on Which These Computations Were Conducted

duration metrics	
average training duration	$\mu_t = 58.662$ h
standard deviation of training duration	$\sigma_t = 1.360$ h
workstation specifications	
CPU	i7-9800X CPU @ 3.80 GHz
RAM	126 GB
hard drive memory	2 TB
OS	Ubuntu 18.04.5 LTS

TABLE IV Considered Hyperparameters With Their Respective Search Spaces and the Found Three Best Parameter Sets

symbol	description	search space	best set	2nd best set	3rd best set
γ	discount factor	$[0, 0.999]$	0.868	0.881	0.849
α_{initial}	initial learning rate	$[1 \cdot 10^{-8}, 1 \cdot 10^{-4}]$	$2.887 \cdot 10^{-5}$	$3.170 \cdot 10^{-5}$	$3.477 \cdot 10^{-5}$
α_{final}	final learning rate	$[1 \cdot 10^{-8}, 1 \cdot 10^{-4}]$	$1.736 \cdot 10^{-5}$	$8.756 \cdot 10^{-5}$	$8.602 \cdot 10^{-5}$
$k_{0\alpha}$	starting step of learning rate reduction	$0, 5 \cdot 10^3, ..., 1 \cdot 10^6$	$4.600 \cdot 10^5$	$3.600 \cdot 10^5$	$9.950 \cdot 10^5$
$k_{1\alpha}$	interval of learning rate reduction	$5 \cdot 10^4, 5.5 \cdot 10^4, ..., 3 \cdot 10^6$	$2.710 \cdot 10^6$	$1.995 \cdot 10^6$	$2.100 \cdot 10^6$
$\epsilon_{\text{initial}}$	initial exploration rate	$[0.0, 0.5]$	$2.119 \cdot 10^{-1}$	$1.797 \cdot 10^{-1}$	$2.553 \cdot 10^{-1}$
ϵ_{final}	final exploration rate	$[0.0, 0.2]$	$1.774 \cdot 10^{-1}$	$1.111 \cdot 10^{-1}$	$1.859 \cdot 10^{-1}$
k_ϵ	interval of exploration rate reduction	$5 \cdot 10^4, 5.5 \cdot 10^4, ..., 3 \cdot 10^6$	$2.210 \cdot 10^6$	$2.160 \cdot 10^6$	$2.445 \cdot 10^6$
l	number of hidden network layers	$1, 2, ..., 10$	10	10	10
n	number of neurons per hidden layer	$20, 40, ..., 1000$	560	540	440
ρ	target network update parameter	$[0.001, 0.999]$ and $100, 200, ..., 5 \cdot 10^4$	$2.096 \cdot 10^{-1}$	$1.767 \cdot 10^{-1}$	$1.425 \cdot 10^{-1}$
$\lvert\mathcal{B}\rvert$	mini-batch size	$8, 16, 32, 64$	32	64	64
$\lvert\mathcal{D}\rvert$	replay buffer size	$5 \cdot 10^3, 1 \cdot 10^4, ..., 5 \cdot 10^5$	$3.650 \cdot 10^5$	$3.950 \cdot 10^5$	$4.000 \cdot 10^5$
f	activation function	see Tab. 5	LeakyReLU	LeakyReLU	LeakyReLU
β	activation function parameter	see Tab. 5	$2.908 \cdot 10^{-1}$	$3.425 \cdot 10^{-1}$	$1.785 \cdot 10^{-1}$
K	maximum episode length	$100, 200, ..., 5 \cdot 10^4$	$1.490 \cdot 10^4$	$0.610 \cdot 10^4$	$1.920 \cdot 10^4$
M	total training steps	$3 \cdot 10^6$ (fixed)			
G	performance metric (20)		67.51 %	67.41 %	67.40 %

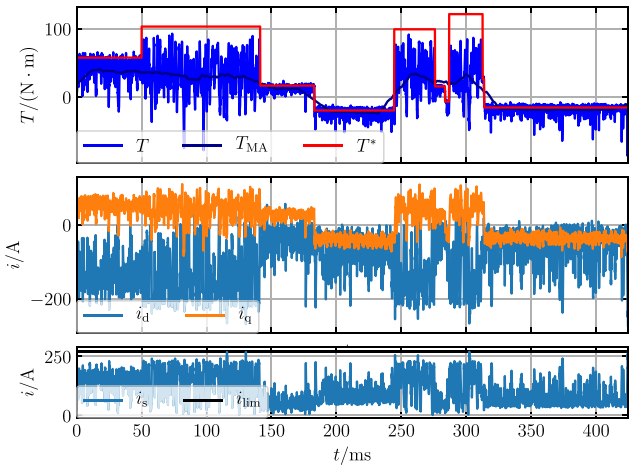

FIG. 8. Exemplary DQ-DTC performance in an intermediate training episode, $\omega_{\text{me}} = 891.291\,\text{s}^{-1}$.

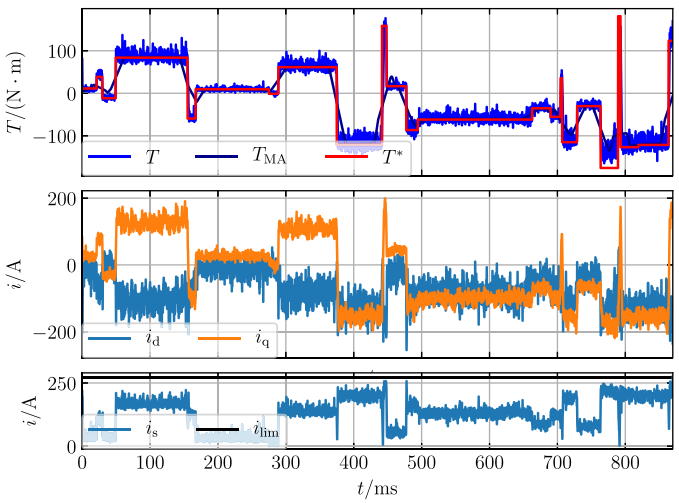

FIG. 9. Exemplary DQ-DTC performance in an advanced training episode, $\omega_{\text{me}} = -155.149\,\text{s}^{-1}$.

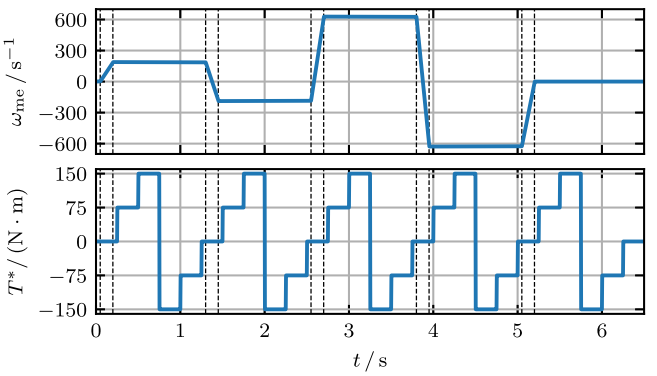

FIG. 10. Controller validation profile.

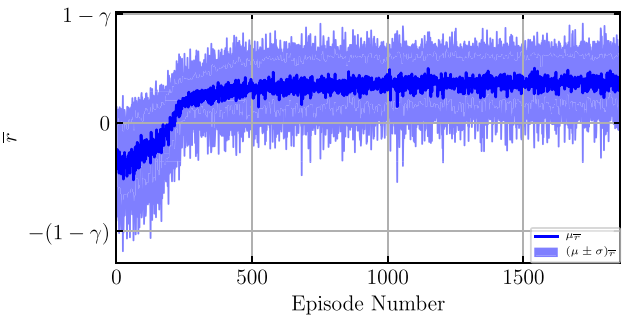

FIG. 11. Convergence behavior of the best found DQ-DTC hyperparameter set from Table II during 50 distinct trainings with averaged mean reward per episode $\mu_{\bar{r}}$ and a confidence interval of one standard deviation $\sigma_{\bar{r}}$.

TABLE V Search Space of the DQN Activation Functions

name	definition	parameters
SoftPlus	$y = \ln(1 + e^x)$	
LeakyReLU	$y = \max(\beta x, x)$	$\beta \in [0, 0.5]$
ELU	$y = \begin{cases} \beta(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$	$\beta \in [0, 10]$
SELU	$y = \lambda \begin{cases} \beta(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$	$\lambda = 1.05070098$ $\beta = 1.67326324$
Sigmoid	$y = \frac{1}{1+e^{-x}}$	
tanh	$y = \tanh(x)$	

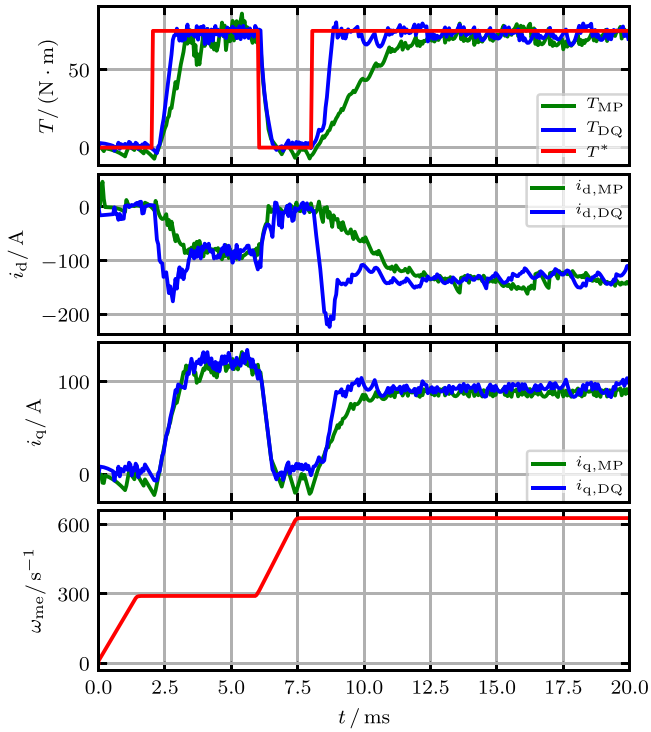

FIG. 12. Reference step performance for the MP-DTC and DQ-DTC.

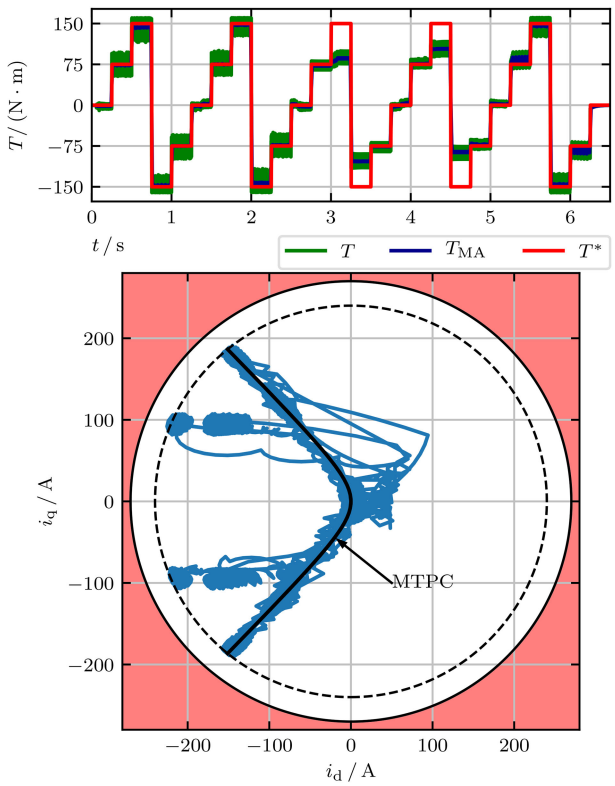

FIG. 13. Validation profile performance for the MP-DTC.

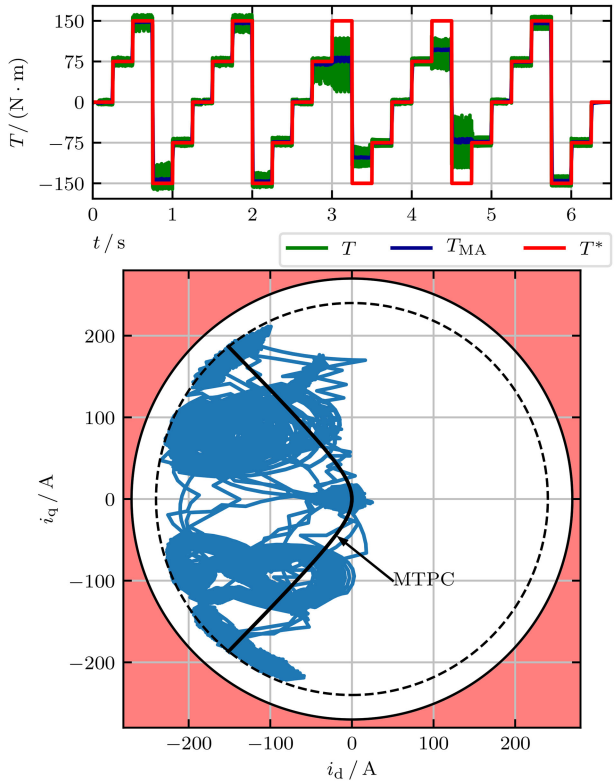

FIG. 14. Validation profile performance for the DQ-DTC.

execution of random actions on the motor environment. Accordingly, the untrained agent is not able to comply to the operation limits and violates them early (Fig. 7). Over the course of the training (Fig. 8), the DQ-DTC agent learns how to track the torque reference T^*, and how limit violations can be avoided more reliably. Finally (Fig. 9), the agent will be able to track the reference torque accurately and with high dynamic. Nevertheless, exploration necessitates occasional execution of random actions and even late training episodes are not entirely free of limit violations.

Further excitation of the control loop is ensured via random changes of the reference torque, as can be seen in Fig. 9. At any time step k there is a chance of 0.1 % that the reference torque is resampled, otherwise it is kept constant. Please note that the maximum reference torque T_{lim} surpasses the nominal torque T_{n} of the considered PMSM, which would require overcurrent operation (regions $\mathbb{D}$ and $\mathbb{E}$). This way, the controller is forced to operate the plant closely to these regions while preventing to enter them in order to avoid the corresponding lower reward, making the stabilization of the plant integral to the training.

The implemented DQ-DTC training algorithm is available in [34] to supplement this article.

B. HYPERPARAMETER OPTIMIZATION

For the given setup, different configurations of the feedforward multilayer perceptron [25] are considered as a choice for the DQN $\hat{q}_{\boldsymbol{\theta}}$. The selection of an expedient hyperparameter set

is in the following analyzed within the scope of an optimization process. The hyperparameters in question with their respective search spaces are outlined in Tabs. IV and V. The hyperopt Python library [35] in conjunction with the integrated tree-structured Parzen estimator [36] is used as optimization tool. The large computational effort of this optimization is managed using the capacities of a high-performance computing system. The hyperparameter optimization aims to maximize the average reward G for the reward function presented in Sec. IV-C, parameterized with $\gamma = 0$ to yield $G \in [-1, 1]$

$$G = \bar{r}_{\gamma=0} = \frac{1}{K} \sum_{i=0}^{K} r_{\gamma=0,i+1}, \tag{20}$$

on the speed and reference torque validation profile as depicted in Fig. 10. A total of 500 different parameter sets have been considered. The three best results of this investigation are presented in Table IV. It can be seen that these parameter sets are very similar, hinting that a quite reliable parameterization can be found in the corresponding region of the hyperparameter search space.

To evaluate the convergence reliability of the training process with the found best hyperparameter set from Table IV, the control-learning procedure is conducted for another 50 agents. The resulting average learning curve is depicted in Fig. 11. As it seems, the found parameter set permits reliable controller training for the torque control task. It should also be noted that the exploring starts at the beginning of each episode incorporate random set points. This adds some degree of noise to the reward distribution of subsequent episodes since the initial motor states might or might not fit well to these set points. Some variation in reward distribution is therefore to be expected.

Within this convergence investigation the training duration, assessed in Table III, includes both, drive simulation as well as deep Q learning. In consideration of the early performance increase visible in Fig. 11, it could be considered to examine a shortened training phase in the future.

C. PERFORMANCE COMPARISON AGAINST MP-DTC

After hyperparameter optimization, the resulting DQN setup is to be compared against an already established state-of-the-art control solution, allowing a reliable performance ranking. The control method which comes closest to the DQ-DTC is the MP-DTC concept from the MPC domain [5], [6]. The utilized MP-DTC parameterization was designed with respect to the performance metric (20) and is also fully available at [34]. It is realized assuming a one-step prediction horizon, which is the usual choice in FCS-MPC [11]. A larger prediction horizon will only yield negligible performance improvements but will strongly add to computational burden [5], [11].

For the MP-DTC, whose validation episode is depicted in Fig. 13, it is visible that i_d is positive during some transients and that the intended current boundary at i_n was violated very

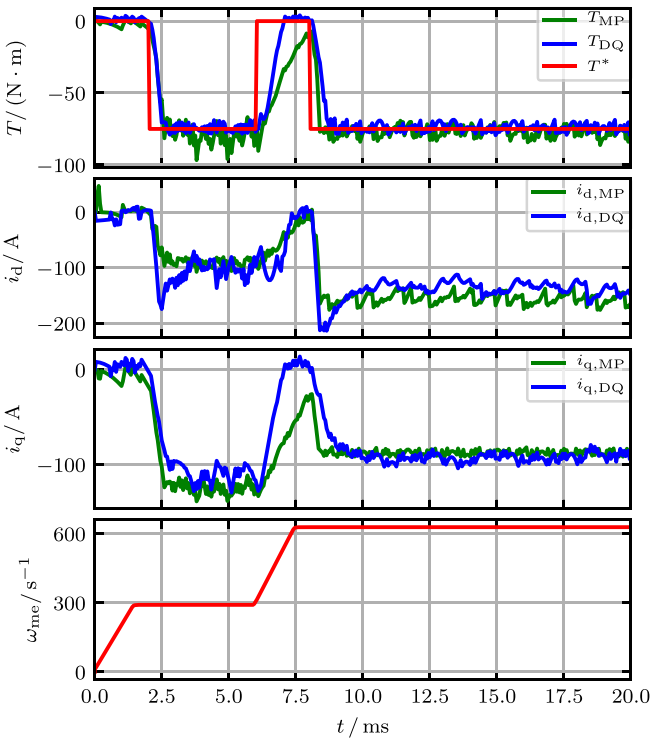

FIG. 15. Reference step performance for the MP-DTC and DQ-DTC.

briefly, which is unwanted behavior, but otherwise the MP-DTC shows very good torque accuracy and efficiency. It is visible that the marked MTPC trajectory is adhered to for most of the time. For higher velocities the MTPC operation points are located outside the ellipsis of available voltage, leading to flux weakening operation.

The validation episode of the DQ-DTC is shown in Fig. 14. This agent is parameterized according to the best set as listed in Table IV. Operating points of positive i_d are less striking in this setup, and moreover, it is visible that the behavior at low speed is preferable in terms of the torque ripple. At higher speed, however, this agent is not as well performing and has problems to steadily hold high torque magnitudes. Interestingly, this seems to be less critical in generator operation ($\omega_\mathrm{me} T < 0$), and more severe in motor operation ($\omega_\mathrm{me} T > 0$).

The transient behavior of both control approaches in reaction to a reference step is highlighted in Fig. 12 and Fig. 15. Here, the DQ-DTC reaches the reference torque faster than the MP-DTC, which is particularly noticeable at higher speed. The average performance of the best ten hyperparameter sets is presented in Fig. 16. This plot suggests that the factor of randomness, that is an inherent part of the training phase, is quite important for the performance outcome. Hence, it is not advisable to use the trained controller without proper testing.

In terms of quantitative performance evaluation, the resulting performance metrics on the validation profile and the step

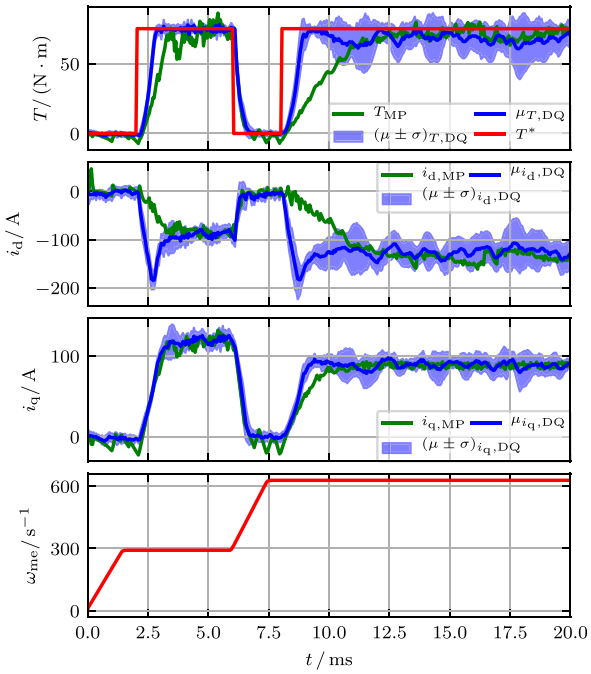

FIG. 16. Reference step performance for the MP-DTC in comparison to the average performance of the best ten DQ-DTC parameterizations with sample mean μ_{DQ} and a confidence interval of one standard deviation σ_{DQ}.

TABLE VI Performance Comparison of MP-DTC and DQ-DTC

experiment	metric	MP-DTC	DQ-DTC
validation profile (Fig. 10, Fig. 13, Fig. 14)	G	62.51 %	67.51 %
	MSE(T)	0.37 %	0.47 %
	MAE(T)	3.32 %	3.16 %
	RMS(i_s)	62.41 %	61.62 %
	$\mathrm{f_{sw}}$	5.99 kHz	6.14 kHz
reference step $T^*>0$ (Fig. 12)	G	61.25 %	69.70 %
	MSE(T)	0.37 %	0.20 %
	MAE(T)	3.51 %	1.89 %
	RMS(i_s)	47.14 %	52.57 %
	$\mathrm{f_{sw}}$	6.67 kHz	6.35 kHz
reference step $T^*<0$ (Fig. 15)	G	62.65 %	69.76 %
	MSE(T)	0.22 %	0.19 %
	MAE(T)	2.32 %	1.77 %
	RMS(i_s)	57.34 %	54.72 %
	$\mathrm{f_{sw}}$	7.45 kHz	6.38 kHz

profile are rated in Table VI. The metrics used are defined by

$$\mathrm{MSE}(T) := \frac{1}{K}\sum_{i=0}^{K}\left(\frac{T_i^* - T_i}{2T_\mathrm{lim}}\right)^2,$$

$$\mathrm{MAE}(T) := \frac{1}{K}\sum_{i=0}^{K}\left|\frac{T_i^* - T_i}{2T_\mathrm{lim}}\right|,$$

$$\mathrm{RMS}(i_\mathrm{s}) := \sqrt{\frac{1}{K}\sum_{i=0}^{K}\left(\frac{i_{\mathrm{s},i}}{i_\mathrm{lim}}\right)^2},$$

$$\mathrm{f_{sw}} := \frac{1}{mcT_\mathrm{s}}\sum_{k=0}^{K}||\Delta \boldsymbol{s}_{\mathrm{abc},k}||_1, \qquad (21)$$

with the average switching frequency $\mathrm{f_{sw}}$, the number of converter half bridges $m=3$, the number of converter levels $c=2$, the duration of the sampling period T_s and $\Delta \boldsymbol{s}_{\mathrm{abc},k} = \boldsymbol{s}_{\mathrm{abc},k} - \boldsymbol{s}_{\mathrm{abc},k-1}$. For the validation profile the torque ripple that was observed for the DQ-DTC at higher speed leads to a higher torque MSE. Furthermore, since these operating points are also characterized by a diminished torque output, the power flux to / from the drive is lowered as well, leading to a lower current RMS. Due to the higher rating concerning the performance metric G it can be expected that the DQ-DTC achieved a higher drive efficiency than the MP-DTC. For the reference steps, the gained metrics fit the observed behavior very well: in both cases, the DQ-DTC shows better dynamic behavior and, hence, better performance concerning the torque reference. The average switching frequency - as an indicator for inverter switching losses - is in the same value range for both considered algorithms. However, the switching frequency is operation point-dependent and was not addressed as part of the optimization task in either control approach, so separate consideration of it should be made in future work in this area.

VI. CONCLUSION AND OUTLOOK

A data-driven PMSM torque controller has been successfully trained using the developed reward function within the deep Q-learning algorithm. The resulting DQ-DTC agent was able to perform comparable to a state-of-the-art MP-DTC controller which had full and precise model knowledge. It has been shown that the RL controller design approach without plant-specific knowledge yields a controller with satisfactory performance. However, since no general stability and performance theory has yet been established in the domain of RL, comprehensive and rigorous controller testing is necessary before deployment. The results of the hyperparameter optimization propose a DQN design configuration that should be reusable for any other SM drive.

In the upcoming research it has to be investigated in how far the same setup can be transferred to a physical test bench drive system. If trained online, the DQ-DTC allows consideration of the usually challenging parasitic effects of (cross-) saturation, inverter nonlinearity, constructive anisotropy and iron losses. Furthermore, switching losses could be incorporated into the design of the presented holistic performance- and efficiency-motivated control algorithm. However, online learning also needs to be conducted with caution, because a too random action selection could result in overcurrent situations triggering emergency shutdowns. Moreover, it has to be investigated whether the proposed DQN architecture is real-time capable on industrial, low-cost control hardware.

Moreover, it has to be analyzed if the utilized reward design can be transferred to the continuous control set scenario, and in how far the data-driven torque control approach is feasible for non-synchronous motors.

REFERENCES

[1] 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.

[2] 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.

[3] J. Holtz and S. Stadtfeld, "A predictive controller for the stator current vector of ac machines fed from a switched voltage source," in *Proc. Int. Power Electron. Conf.*, 1983, pp. 1665–1675.

[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, 2020.

[5] M. Preindl and S. Bolognani, "Model predictive direct torque control with finite control set for PMSM drive systems, Part 1: Maximum torque per ampere operation," *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 1912–1921, Nov. 2013.

[6] M. Preindl and S. Bolognani, "Model predictive direct torque control with finite control set for PMSM drive systems, Part 2: Field weakening operation," *IEEE Trans. Ind. Informat.*, vol. 9, no. 2, pp. 648–657, May 2013.

[7] 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.

[8] C. Wolz, "Evaluation of parameter variation and torque accuracy of ipmsm for ev applications," in *Proc. Int. Conf. Elect. Mach.*, vol. 1, 2020, pp. 1335–1341.

[9] M. Stender, O. Wallscheid, and J. Böcker, "Development of a black-box two-level igbt three-phase inverter compensation scheme for electrical drives," in *Proc. IEEE 28th Int. Symp. Ind. Electron.*, 2019, pp. 296–301.

[10] W. Kirchgässner, O. Wallscheid, and J. Böcker, "Deep residual convolutional and recurrent neural networks for temperature estimation in permanent magnet synchronous motors," in *Proc. IEEE Int. Elect. Mach. Drives Conf.*, 2019, pp. 1439–1446.

[11] P. Karamanakos and T. Geyer, "Guidelines for the design of finite control set model predictive controllers," *IEEE Trans. Power Electron.*, vol. 35, no. 7, pp. 7434–7450, Jul. 2020.

[12] P. Karamanakos, T. Geyer, and R. P. Aguilera, "Long-horizon direct model predictive control: Modified sphere decoding for transient operation," *IEEE Trans. Ind. Appl.*, vol. 54, no. 6, pp. 6060–6070, Nov./Dec. 2018.

[13] T. Dorfling, H. du Toit Mouton, T. Geyer, and P. Karamanakos, "Long-horizon finite-control-set model predictive control with nonrecursive sphere decoding on an FPGA," *IEEE Trans. Power Electron.*, vol. 35, no. 7, pp. 7520–7531, Jul. 2020.

[14] T. Geyer, "Computationally efficient model predictive direct torque control," *IEEE Trans. Power Electron.*, vol. 26, no. 10, pp. 2804–2816, Oct. 2011.

[15] D. Silver, "Reinforcement learning - lecture notes," 2015. [Online]. Available: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

[16] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[17] W. Kirchgässner, M. Schenke, O. Wallscheid, and D. Weber, "Reinforcement learning course material," 2020, Paderborn University. [Online]. Available: https://github.com/upb-lea/reinforcement_learning_course_materials

[18] A. Traue, G. Book, W. Kirchgässner, and O. Wallscheid, "Toward a reinforcement learning environment toolbox for intelligent electric motor control," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–10, 2020. doi: 10.1109/TNNLS.2020.3029573.

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

[20] G. Book *et al.*, "Transferring online reinforcement learning for electric motor control from simulation to real-world experiments," *IEEE Open J. Power Electron.*, pp. 1–1, 2021. doi: 10.1109/OJPEL.2021.3065877.

[21] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature - Int. J. Sci.*, vol. 518, no. 7540, pp. 529–533, 2015.

[22] R. De Doncker, D. Pulle, and A. Veltman, *Advanced Electrical Drives: Analysis, Modeling, Control*. Cham, Switzerland: Springer Nature Switzerland AG, vol. 52, Jan. 1, 2011.

[23] 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, p. 2498, 2021, doi: 10.21105/joss.02498.

[24] R. Bellman, "On the theory of dynamic programming," *Proc. Nat. Acad. Sci. USA*, 1952.

[25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, .

[26] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 30, no. 1, Mar. 2016. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/10295

[27] 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.

[28] C. M. Hackl, J. Kullick, H. Eldeeb, and L. Horlbeck, "Analytical computation of the optimal reference currents for MTPC / MTPA, MTPV and MTPF operation of anisotropic synchronous machines considering stator resistance and mutual inductance," in *Proc. 19th Eur. Conf. Power Electron. Appl.*, 2017P.1-P.10.

[29] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford Univ. Press, Clarendon Press, 1995.

[30] M. Plappert, "Keras-Rl," 2016. [Online]. Available: https://github.com/keras-rl/keras-rl

[31] T. McNally, "keras-Rl2," 2019. [Online]. Available: https://github.com/wau/keras-rl2

[32] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, 2016. [Online]. Available: http://arxiv.org/abs/1603.04467

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, San Diego, 2017.

[34] M. Schenke and O. Wallscheid, "Supplementary material: DQ-DTC training and validation code notebooks," 2020, [Online]. Available: https://github.com/max-schenke/DQ-DTC

[35] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," Ser. *Proc. Mach. Learn. Res.*, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 1, pp. 115-123. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013. [Online]. Available: http://proceedings.mlr.press/v28/bergstra13.html

[36] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proc. 24th Int. Conf. Neural Inf. Process. Syst.,* Red Hook, NY, USA: Curran Associates Inc., 2011, pp. 2546–2554.

MAXIMILIAN SCHENKE received the bachelor's and master's (Hons.) degrees in electrical engineering from Paderborn University, Paderborn, Germany, in 2017 and 2019, respectively. He is currently working toward the Doctoral degree in electrical engineering with the Department of Power Electronics and Electrical Drives. His research focuses on the application of reinforcement learning methods in the domain of drive control.

OLIVER WALLSCHEID (Member, IEEE) received the bachelor's and master's degrees (Hons.) in industrial engineering and the Doctorate degree (Hons.) in electrical engineering from 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. Currently he is also serving as an acting Professor of the Automatic Control Department at Paderborn University. His research interests include data-driven identification and intelligent control of electrical power systems in decentralized grids, power electronics and drives.