Using AI for Closed-Loop Control of a Buck Converter Application

Marvin Slippens[1], Christian P. Dick[1]

[1] TH Köln - University of Applied Sciences, Germany

Corresponding author: Christian P. Dick, christian.dick@th-koeln.de

Abstract

This paper addresses the training of an artificial intelligence (AI) to control the transistor state of a buck converter the most efficient way. For this purpose, a buck converter simulation script was implemented in python. Afterwards, a reinforcement learning agent was trained to provide the optimal duty cycle for the PWM. The trained agent was then tested in the python environment.

1 Introduction

The increasing significance of AI for optimization tasks found an useful application field in DC converters, which show non-linear behaviour. The project objective is to train an AI to provide optimal duty cycles for the backend PWM to obtain a fast and accurate closed-loop control for a buck converter as shown in Fig. 1. The transient state of a buck converter shows non-linear behaviour with regards to two discrete switching states and possibly discontinuous conduction mode. When implementing its control strategy for the output voltage u_R , there is a trade-off between a short rise time and the overshoot of the voltage. Reinforcement learning, as a sub-field of AI is predestined to solve for the optimal solution of this non linear problem field. Initially, a python simulator for the ideal buck converter behaviour was implemented. The AI was configured and trained in this environment to obtain an understanding of the system's interdependencies.

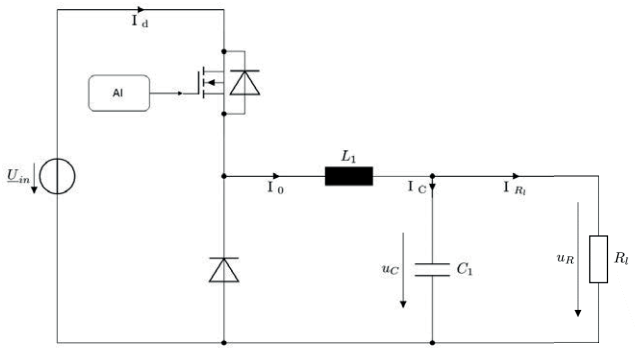

Fig. 1: Buck Converter

2 Reinforcement Learning

Reinforcement learning (RL) is a subfield of machine learning, which is a subfield of AI. The basic idea is to have a so-called agent, who takes actions in an environment. The environment responds to this actions with giving the agent information about its state. According to the agent's task, the state information is used to tweak its strategy. The basic principle is shown in Fig. 2.

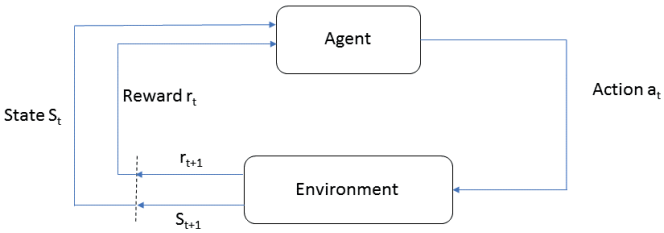

Fig. 2: Reinforcement principle

ISBN 978-3-8007-5245-4

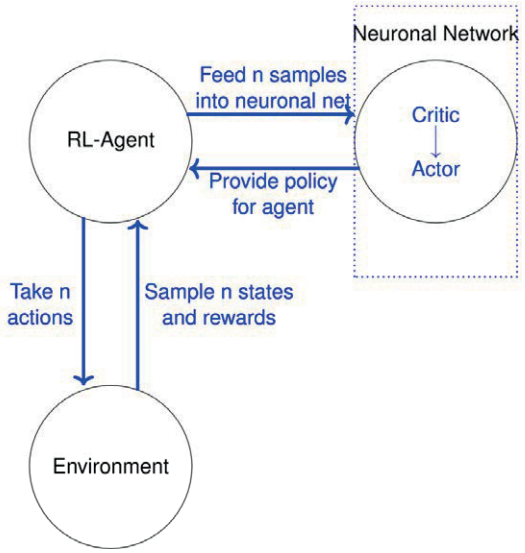

Fig. 3: Advantage Actor Critic

This project uses the A2C (Advantage Actor Critic) reinforcement learning algorithm to train the agent. The main principle of the A2C is shown in Fig. 3. The RL-agent performs n actions in the environment under current policy and samples the n states and rewards. These samples, are fed into the neuronal network, which consists of an actor and a critic. Backpropagation is used as a learning technique to tweak the parameters of the neurons such as weights and biases. In this way the state value function $V^{\pi}(s)$ will be parametrized by the neuronal network (Critic). While training, the estimated value of the state value function will approach the real value of the state value function. With the state value function the advantage will be calculated. The actor uses the estimated advantage of the critic to adjust its policy via a policy gradient method [1].

3 Python simulator

For the training part of the agent an environment was set up in python. It simulates the behaviour of an ideal buck converter. Each action of the agent will lead to an evaluation of the following state space representation: Eq. (1) is used to compute the ON state, while eq. (2) describes the OFF state with activated diode and eq. (3) shows the case with switch and diode both turned OFF.

$$\frac{d}{dt}\begin{pmatrix} I_0 \\ u_R \end{pmatrix} = \begin{pmatrix} 0 & -L^{-1} \\ C^{-1} & -(CR)^{-1} \end{pmatrix}\begin{pmatrix} I_0 \\ u_R \end{pmatrix} + \begin{pmatrix} L^{-1} \\ 0 \end{pmatrix} \cdot U_{in} \quad (1)$$

$$\frac{d}{dt}\begin{pmatrix} I_0 \\ u_R \end{pmatrix} = \begin{pmatrix} 0 & -L^{-1} \\ C^{-1} & -(CR)^{-1} \end{pmatrix}\begin{pmatrix} I_0 \\ u_R \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} \cdot U_{in} \quad (2)$$

$$\frac{d}{dt}\begin{pmatrix} I_0 \\ u_R \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & -(CR)^{1} \end{pmatrix}\begin{pmatrix} I_0 \\ u_R \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} \cdot U_{in} \quad (3)$$

The python simulator acts as the environment which is depictetd in Fig. 2. As input it takes the action from the agent a_t, namely the duty cycle. In accordance with the reward function denoted in table 1 the reward r_{t+1} is calculated. This reward and the result of the simulation is then passed as a tuple (S_{t+1}, r_{t+1}) to the agent.

4 Networks' architecture and configuration

The actor's network architecture is shown in Fig. 4. The critic's network architecture is displayed in Fig. 5. The blocks in these figures represent the different layers. The input layers are made up of two neurons, representing the two dimensional input data. The question marks represent the number of samples which are fed into the layers at once. The hidden layers consists of 100 neurons. Table 1 shows the hyperparamter and additional information of the reinforcement learning process. The parameters of the buck converter, which are used in the python simulator, are summarized in table 2. The hidden layers use *relu* as activation function. The output layer of the critic uses a *linear* activation function. The **mu** output of the actor network uses *tanh* for activation whereas the **sigma** output also uses *linear* activation function. In each step the current pair of (output voltage, inductor current) are fed into the first layer of the actor and critic network. The layers are all feed-forward connected. The actor network features two output neurons, namely mu and sigma. These are used to compute a normal distribution. Afterwards a random sample of this distribution is drawn, which is then used as the action. The critic outputs its estimation of the reward and is only used for updating the weights and biases of the neuronal net. The reason why two hidden layers are used is due to the fact that neuronal networks with two hidden layers can

ISBN 978-3-8007-5245-4

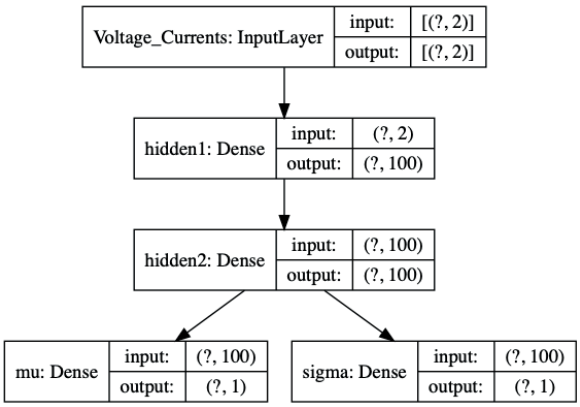

Fig. 4: Actor model architecture

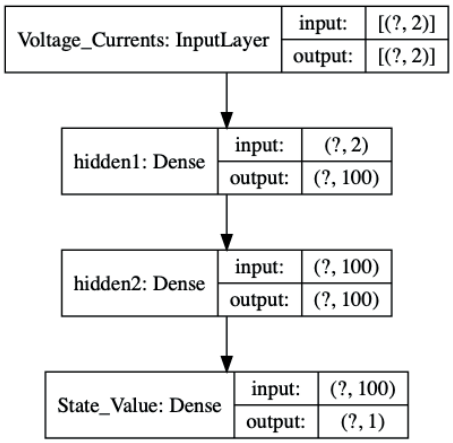

Fig. 5: Critic model architecture

represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy [2].

5 Training and testing results

The training goal for the agent was to control the mean output voltage of the buck converter u_R to 12 V with 24 V as input voltage. After finishing each epoch the voltages and currents of the simulation were set to zero. This forces the agent to pass the transient phase periodically. Fig. 6 shows the reward which was obtained during the training process. In each step the maximum reward is one which can be concluded by the reward function in table 1. One epoch consists of 200 steps which leads to an optimal reward value of 200 per epoch when summing up the rewards. This value is approached during the training. In Fig. 7 the courses of the output voltage, the inductor current and the duty cycles are presented. It can be stated

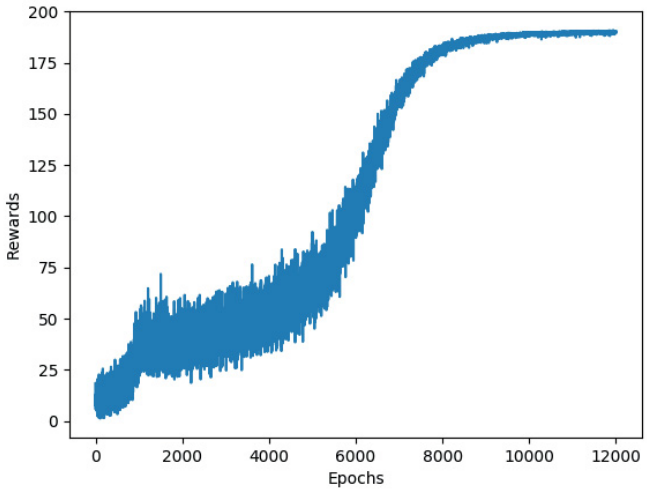

Fig. 6: Reward progression over 12000 epochs

that the output voltage retains at the target output voltage of 12 V after the transient phase. The inductor current holds its analytical steady state value of 2.4 A accordingly. The reason for this course of voltages and currents has its origin in the actions being taken by the agent. The agent chooses 0.5 as duty cycle for the steady state, which is the analytical optimum. The main focus of interest in this project lies in control of the non-linear transient phase of the buck converter. Fig. 9 shows the output voltage during the transient phase. The peak overshoot is 12.3363 V which is equivalent to an overshoot of 2.8 %. The actions which led to this outcome as well as the transient course of the inductor current are displayed in Fig. 8. The steady state is reached after 170 µs which equals 17 periods.

6 Conclusion

During the project, the buck converter simulation script was implemented using python. Furthermore, the OpenAI Gym interface was set up and the RL-agent was coded from scratch. The results show that the agent learnt to control the output voltage u_R to meet the target output voltage. The actions taken by the agent can be classified as optimal during the steady state as they met the analytical optimal value. When analyzing the transient behaviour one can argue that there is an overshoot of the voltage. Throughout the development of the agent other results without an overshoot were produced. In

ISBN 978-3-8007-5245-4

Tab. 1: Parameter used in training

Configuration	
Parameter	Value
Maximum epochs	6000
Length of epoch	200
Gamma	0.9
Actor learning rate	1e-6
Critic learning rate	1e-6
Batch size	32
Optimizer	Adam
Reward function	$e^{-\lvert target\_output\_voltage - actual\_output\_voltage \rvert}$

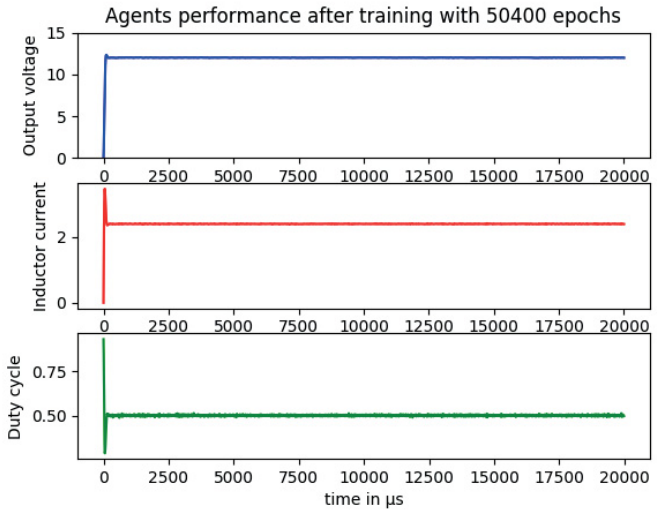

Fig. 7: Course of output voltage, inductor current and duty cycles

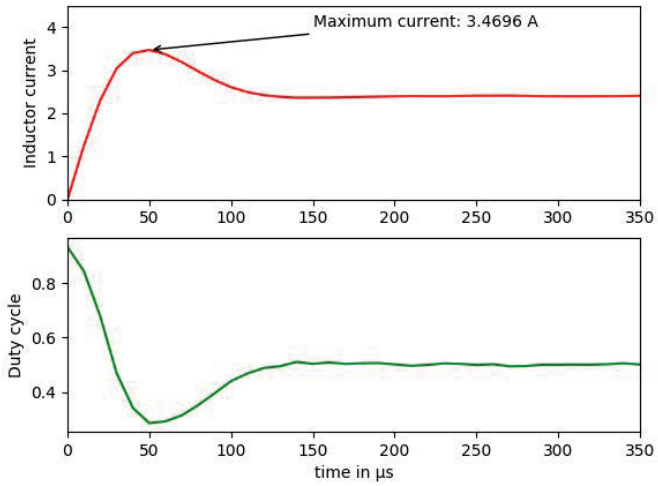

Fig. 8: Transient course of inductor current and duty cycles

Tab. 2: Parameter of buck converter simulator

Configuration	
Parameter	Value
U_{in}	24 V
R_l	5 Ω
L_1	180 µH
C_1	10 µF
Switching frequency	100 kHz

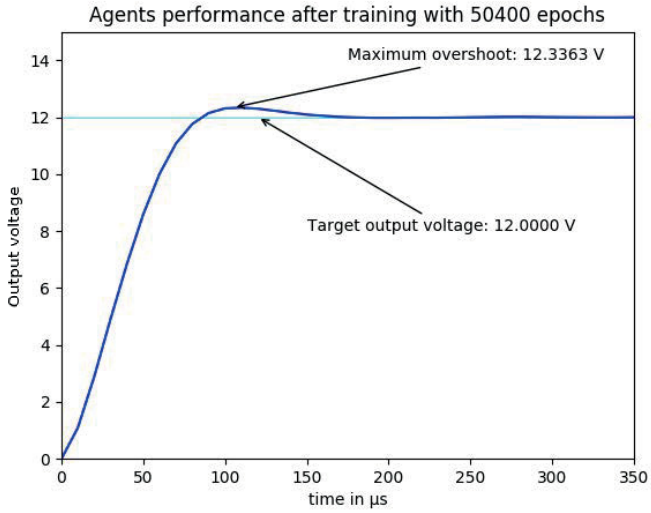

Fig. 9: Transient phase of the output voltage

direct comparison with the here displayed agent it can be stated, that the reward here was higher. This concludes that accepting an overshoot to meet the target output voltage faster yields in a smaller sum of deviations from the target output voltage, which was the goal of the agent from the very beginning.

7 Outlook

The current work shows the possibilities of using AI. Especially using algorithms which support continuous action spaces can be beneficial for finding optimal solutions. One aspect to be followed is to train the agent to control the output voltage during changes of the load or input voltage. After training the agent to cope with major influences which occur in real life systems, we expect a trained AI featuring high performant transient behaviour, being visible in time dependent duty cycles leading to a short rise time of the target output voltage while maintaining high accuracy with low overshoot

ISBN 978-3-8007-5245-4

as well as controlling the output to the target value during disruptions and changes of circumstances. Finally the AI will be ported to a microcontroller for testing on hardware.

References

[1] D. John Pradeep, Mathew Mithra Noel, N. Arun, ”Nonlinear control of a boost converter using a robust regression based reinforcement learning algorithm”, sciencedirect, https://doi.org/10.1016/j.engappai.2016.02.007

[2] Jeff Heaton, ”The Number of Hidden Layers”, https://www.heatonresearch.com/2017/06/01/hidden-layers.html

ISBN 978-3-8007-5245-4