Automation in Construction 127 (2021) 103701

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Reinforcement learning based process optimization and strategy development in conventional tunneling

Georg H. Erharter [a,*], Tom F. Hansen [b], Zhongqiang Liu [b], Thomas Marcher [a]

[a] Graz University of Technology, Institute for Rock Mechanics and Tunnelling, Rechbauerstraße 12, Graz, Austria
[b] Norwegian Geotechnical Institute, Oslo, Norway

ARTICLE INFO

Keywords:
Conventional tunneling
Reinforcement learning
Tunnel excavation strategy
Machine learning
Excavation sequences

ABSTRACT

Reinforcement learning (RL) - a branch of machine learning - refers to the process of an agent learning to achieve a certain goal by interaction with its environment. The process of conventional tunneling shows many similarities, where a geotechnician (agent) tries to achieve a breakthrough (goal) by excavating the rockmass (environment) in an optimum way.

In this paper we present a novel RL based framework for strategy development for conventional tunneling. We developed a virtual environment with the goal of a tunnel breakthrough and with a deep Q-network as the agent's architecture. It can choose from different excavation sequences to reach that goal and learns to do so in an economical and safe way by getting feedback from a specially designed reward system. Result analyses show that the optimal policies have great similarities to current practices of sequential tunneling and the framework has the potential to discover new tunneling strategies.

1. Introduction

Digitalization in tunneling is an ongoing process including topics like Artificial Intelligence (AI) technology [1], Building Information Modelling [2] or embedded strain measurements in shotcrete lining [3]. Machine Learning (ML) applications in particular have so far mostly focused on classification tasks based on supervised ML: e.g. [4–6] apply artificial neural networks (ANN) for rockmass behavior classification of tunnel boring machine (TBM) data; [7] use supervised learning methods for automatic work progress identification in NATM (New Austrian Tunneling Method) tunneling; see [8–11] for state-of-the-art reviews on this topic. Applications of unsupervised ML are fewer in number and often related to the search for less dependency on labelled datasets or more objectiveness (e.g. [12,13]).

By the time of this writing reinforcement learning (RL) is mostly a matter of research, but already shows mature problem-solvers for game like scenarios [14,15]. Currently, there is a transition from academia to real-world prototypes, with RL-examples like the optimization of a manufacturing process [16], for real-time steering of hydrocarbon drilling [17,18], in optimization of power grids [19] and control systems in general, for AUVs [20], and in robotics and other autonomous vehicles. To our knowledge, however, there is no published application of RL to geotechnics in general or tunneling in particular.

Today's conventional tunneling – sometimes referred to as "drill and blast tunneling" or "sequential excavation method" – is the classical way of tunnel construction and is the product of more than a century of engineering experience [21]. Great flexibility to adapt to changing ground conditions is one of the main benefits of this type of tunneling, but technical and economic success of the excavation is dependent on the experience of the involved engineers and workers. Albeit experience is undoubtedly valuable, depending on it sometimes goes along with simple repetition of "proven ways" or even negligence of innovation. Furthermore, developments in conventional tunneling often have a strong connection to their nation of origin (e.g. Austria: New Austrian Tunnelling Method [22], Norway: Norwegian Tunneling Method [23], Italy: New Italian Tunneling Method [24,25], etc.) which raises concerns about biased researchers and engineers. The goal of this study is to take a first step in the direction of a conceptual RL-model for optimum decision making in conventional tunneling that is as free as possible of conservatism and national biases. Furthermore, as shown in the development of the RL-agent AlphaGO [15], RL systems have the potential to find new solutions to old problems and thus discover unimagined strategies.

In this paper we present a novel RL based framework for construction process optimization and strategy development for conventional

* Corresponding author.
E-mail address: erharter@tugraz.at (G.H. Erharter).

https://doi.org/10.1016/j.autcon.2021.103701
Received 1 December 2020; Received in revised form 29 March 2021; Accepted 2 April 2021
Available online 17 April 2021

tunneling. Such models can act as decision support for the geotechnical engineer, engineering geologist, geotechnician etc. (hereafter "geotechnician" is used) (design choices, progress-planning) and in the long run such models work towards full automation in underground construction. Hence, the model is a first attempt to automate decisions made by the geotechnician on face in underground construction.

In the next section (2) we frame the process of conventional tunneling as a RL problem and provide details on how these two disciplines can connect. Section 3 is the main methodological section that presents the geotechnical scenario at the background of this RL simulation as well as the agent and the environment. In Section 4 we describe the training process and in Section 5 we show experiences gathered during the training and testing of agents. A conclusion and implications for the vision of "digital tunneling" is given in Section 6 and we present an outlook in the last Section 7. A reference to the Python based code for this paper, is given in the appendix.

2. Conventional tunneling as a reinforcement learning problem

The process of reinforcement learning (RL) is typically depicted as a closed loop where an agent takes different actions, to influence an environment which responds by sending an updated state as well as a reward signal to the agent (e.g. [26]). To apply RL, the learning problem must undergo the Markov property, i.e. we only need to know the current state of the system, to make a decision [27]. Therefore, the state must include information about all aspects of the past agent–environment interaction that make a difference for the future. This can be said to be true for tunnel excavation where we only need to know the state of the rockmass and the excavation-process to decide on how to proceed.

Tunnel construction follows several cyclic and sequential processes, some of which can be framed as loops and therefore translated to RL problems. The most outstanding loop in conventional tunneling is the excavation of an underground opening with a sequential construction process of: blasting, mucking and rock support installation [28] ("excavation loop" in Fig. 1). While the components of this excavation loop are themselves often sequential processes (e.g. blasting sequence, support installation etc.), the excavation loop specifically is based on a sequence of geotechnical decisions belonging to a bigger cycle which we refer to as the excavation sequence decision – loop. Looking at the bigger picture, excavation sequence decisions are one part of the whole construction phase of a tunnel and therefore part of the whole tunnel life cycle (Fig. 1).

The focus of this study is to create a simulation of a simplified version of the process that governs the general excavation sequence decisions and frame it as a RL-loop. This process can be translated to a RL-loop/Markov decision process [27] consisting of the following components:

- the decision making geotechnician is the **agent**
- processes like "top heading excavation", "bench excavation", "installation of face support" etc. are the **actions**
- the rockmass itself and the construction site with all its processes are the **environment** (here described at each timestep by the state of the environment and the reward-system)
- the sum of all delays (planned and unplanned) and complications throughout the course of the excavation which result from the geotechnician's actions are the **reward**
- the current state of the excavation including information about the past and recent rockmass conditions as well as the already installed support are the **state**

Fig. 2 is a graphical representation of this process which we refer to as "TunnRL" (Tunnel automation with Reinforcement Learning). The individual components of this loop in the above given list as well as in Fig. 1 and Fig. 2 are only for explanatory purpose and do not claim to be complete. We give detailed lists of the possible actions in Section 3.2.1 and explain the exact content of the state and the rewards in the Sections 3.3.1 and 3.3.2 respectively.

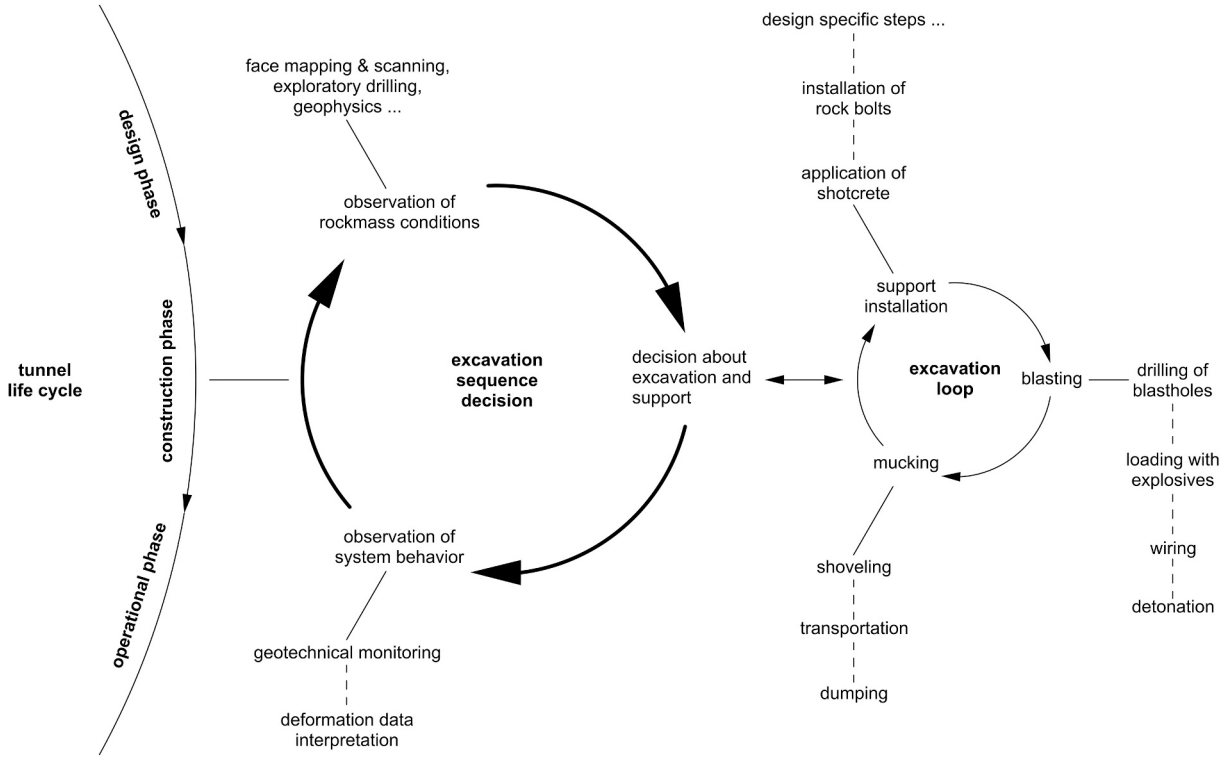

Fig. 1. Schematic diagram of cyclical and sequential processes in conventional tunnel excavation. The size of loops represents a qualitative hierarchy with smaller loops being components of bigger loops. The excavation sequence decision – loop is given in bold, as this will be the main focus of this paper.

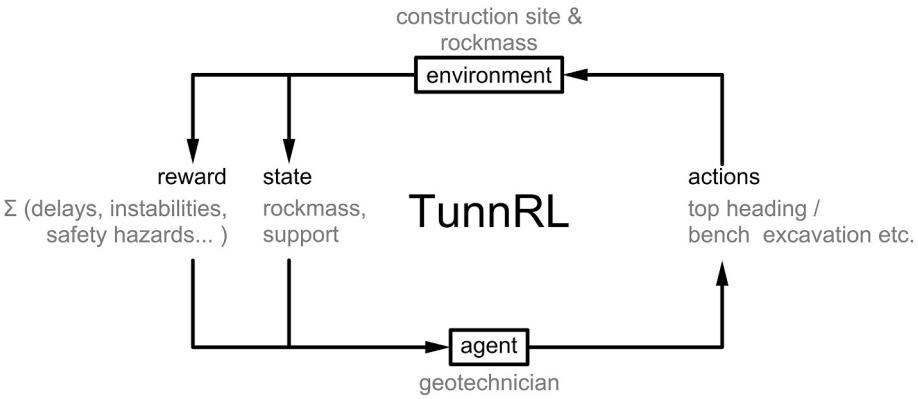

Fig. 2. Simplified schematic plot of components of drill and blast tunneling (grey) as a reinforcement learning process - TunnRL.

2.1. Simplifications

As shown in Fig. 1, conventional tunneling is a complex process that consists of several subprocesses which again have subprocesses etc. Attempting to formulate this whole system as a RL problem with all involved details is out of the scope of an initial study which should serve as the basis for future research. Additionally, it must also be considered that RL is still at the very beginning of practical applications. Consequently, we had to take several simplifications of conventional tunneling to reduce the general complexity of the simulation and the size of the state - and action space. Still, we consider modelling the major decision-loop and the actions and rewards, not to be too far from a realistic tunnel approach, and will contribute with important insights to the use of RL in the optimization of the conventional tunnel cycle.

Some important simplifications in comparison to reality were made in this study:

- Available partial excavation methods are restricted to either top heading - or a combined bench and invert excavation (described as "bench excavation" hereafter) with a specific tunnel geometry.
- Already excavated parts are considered to be stable and the process of tunnel lining installation is not dealt with in the simulation.
- If installed, support ahead of the face always consists of 10 m long face anchors. The supported area ahead is stable and excavation within that stable area cannot lead to failure. No other types of rock support are considered, such as radial bolts and shotcrete.
- We evaluate if stable conditions are at hand in the excavation area by the face pressure equation for open face tunneling after [29] (see Eq. (3) in Section 3.3.2). We chose this analytical solution as it is a computationally efficient way for a stability assessment that tells if stability is given or not at a certain ground type. Although this is only one aspect of the tunnel stability considerations, we see this approach as sufficient for the present initial study.
- The stability assessment considers the cross-sectional area only and no longitudinal effects.
- Rockmass quality is reduced to few mechanical values representing "favorable" or "unfavorable" rockmass conditions.
- There are only two available advance lengths (i.e. the length of one blasting round/round of excavation) with 2 and 4 m each.
- Information from probe drilling or deformation monitoring is not simulated and used in the decision process. That means that the agent has no information about the rockmass ahead of the face or no information about eventual deformation behind the face.

3. A simulation of conventional tunneling

As the main goal of this study is to train a RL agent to execute an excavation sequence as efficiently as possible, we designed a simulation of such a scenario. The simulation consists of a longitudinal tunnel section of a specific length (*tl*), where two different types of ground conditions can occur – one favorable and one unfavorable. Before the excavation, the agent is unaware of the distribution of ground types and the distribution of the ground types is only revealed by the excavation itself. The ultimate goal of the agent is to achieve a breakthrough of both the top heading and the bench of the tunnel or in other words, the position of the top heading excavation (pos_{th}) and the position of the bench excavation (pos_{bi}) must be greater than, or equal to *tl*. During the excavation the agent can choose from different actions, e.g. top heading excavation with 2 m advance length, bench excavation with 4 m advance length and installation of face support etc.

3.1. Geotechnical scenario

For this study's simulation we have chosen the following tunneling scenario: The total length of the tunnel (*tl*) is 200 m, as this does not lead to an excessively large state space in the RL model (see Section 3.3.1) but is still a realistic length. The tunnel's cross section has a total area of 91.31m^2 and a height of 10 m with 58.56m^2 and 32.75m^2 being the areas of the top heading - and the bench and invert excavation respectively (see Fig. 3). From these areas, equivalent diameters (*D*, i.e. the diameter of a circle with that area) of 8.41 m and 6.46 m can be computed for both parts of the excavation.

There is a penalty if the distance between the top heading's- and the bench's tunnel face is too big. This distance ($dist_{max}$) is set to be 50 m and the idea behind this is, that in many real projects – especially in soil conditions and long tunnels -, the top heading cannot be driven indefinitely long ahead of the bench, because of safety reasons, necessities to

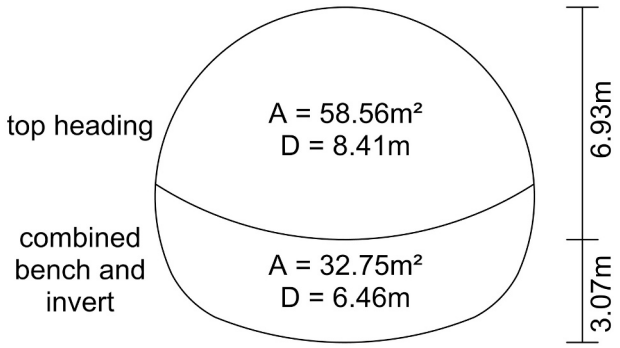

Fig. 3. Tunnel cross section of the given simulation.

have a fast final ring closure, excavate cross cuts or general construction logistics.

We defined two ground types (gt) for the simulation and the relevant ground properties are given in Table 1. Both are considered to be of homogeneous, isotropic and continuous nature and one can imagine them as a type of hard soil/soft rock (HSSR) material [30,31]. With the chosen parameters, gt1 represents "unfavorable" ground conditions and gt2 represents "favorable" ground conditions. The condition for a gt to be favorable/unfavorable is based on the ground properties in combination with the given tunnel geometry (see above) as evaluated by the chosen stability assessment criterion (see Section 3.3.2). The simulated tunnel is situated above the groundwater table and the permeability is set to 10^{-5} m/s for both gt.

Choosing the given properties to define the gt, is closely connected to the way the stability assessment of the excavation is done (see Eq. (3) in Section 3.3.2). The permeability of 10^{-5} justifies the use of Eq. (3) for a stability assessment as according to [32] conditions are considered to be drained when the permeability is above 10^{-7} to 10^{-6} m/s. Nevertheless, if ground types with lower permeability/undrained conditions are to be used for the simulation, appropriate solutions for stability assessments must be chosen. Implementing more sophisticated stability assessments that take phenomena like ground water conditions into account is desirable but must be done with care as this heavily influences the overall performance of the tunneling simulation and the RL itself. The outlook and discussion of stability assessments in Section 7 goes into more detail on this topic.

Using random walks with barriers [33], we created unique, 210 m long geological sections with a decimeter resolution (see Section 3.3.1 for why the sections are 210 and not 200 m long). We created a ground type-vector (gt-vector) of 2100 datapoints by scaling the random walk between 0 and 1, rounding to full numbers and using 0 as gt1 and 1 as gt2 (see Fig. 4 bottom row). To transform the gt-vector to a full geological section with one row for the top heading and one for the bench excavation, the vector is horizontally duplicated to an array of 2 $\times$ 2100 datapoints. + 1 is then added to the array so that the number 1 represents gt1, number 2 gt2 and number 0 represents the unexcavated part of the tunnel. The top row of Fig. 4 shows a visualization of such an array, where pos_{th} and pos_{bi} are at 165 and 125 m respectively.

Given the complexity of simulating the process of conventional tunneling (see Section 2), we chose not to complicate the scenario by introducing more gt. However, increasing the number of gt can easily be done with the above described random walk based approach. For example, if it was necessary to simulate four gt then the values of the random walk must be split into four within the boundaries: $gt1 < 0.25$, $0.25 \leq gt2 < 0.5$, $0.5 \leq gt3 < 0.75$ and $0.75 \geq gt4$.

3.2. Agent

Translating the above described geotechnical scenario to RL, the geotechnician who observes the state of the construction and rockmass behavior and makes decisions based on this information, now becomes the "RL agent". Due to the state-complexity of the problem, we chose a deep Q-network (DQN) as the RL agent. Deep Q-learning is a deep reinforcement learning technique that extends the capabilities of classical Q-learning [34,35] by replacing the value iteration in the Q-table with the function approximator of deep artificial neural networks (ANN). Although applications of ANNs for geotechnical purposes are still often seen as complementary to conventional computational models [36,37], in this case the use of ANNs allows for applications in complex and continuous states spaces while classical Q-learning is confined to discrete states.

Table 1
The mechanical parameters and permeability of the two ground types, where gt1 represents weak rock and gt2 stronger rock.

Ground type	Specific weight [kN/m^3]	Cohesion [kPa]	Friction angle [°]	Permeability [m/s]
gt1	24	23	20	10^{-5}
gt2	25	40	30	10^{-5}

DQN algorithms are off-policy, model free RL techniques following the Bellman equation (Eq. (1) after [14]) where the optimal action-value function $Q^*(s,a)$ is based on a state s and after having taken an action a. The best action is chosen by maximizing the expected value of $r + \gamma Q^*(s',a')$ where r is the reward, γ is the discount factor that determines how important the future reward is to the algorithm (see also Table 2), s' is the state at the next time step and a' are all possible actions [14].

$$Q^*(s,a) = \mathbb{E}_{s'}\left[r + \gamma \underset{a'}{max} Q^*(s',a') \,|s,a\right] \tag{1}$$

[14] have shown that the DQN algorithm can be used for a wide range of different RL problems and albeit there are improvements to deep Q-learning we chose the original implementation as we see it as well suited to establish a baseline for further developments in geotechnical RL. Our implementation is based on the DQN after [14] and the custom DQN implementation from [38]. The main deviations of the network architecture in comparison to [14,38] are due to the shape of the input and output data with the input being an 2 $\times$ 2100 $\times$ 2 array and the output a vector of length 8. The number of hidden layers and the decreasing kernel size from the top to the bottom convolutional layers is in accordance with [14] as is the size of the kernel's stride which is half the size of the kernel itself. While [14] used 32, 64 and 64 filters for each of the three hidden layers respectively, we used 32, 64 and 32 filters for these layers, as we observed that the agent's performance did not suffer from this decrease, while the computational speed increased. In accordance with these authors we used rectified linear units (ReLU) [39] activation functions. ReLU activation functions have been widely adopted for ANNs within the past decade as they have shown to achieve a better performance than previously used activation functions like the sigmoid (see [40]). As given in [14] the general DQN's ANN architecture is that of a deep convolutional neural network [41] whose hierarchical structure mimics the effect of receptive fields and is inspired by [42]. Like [14] we did not perform systematic hyperparameter tuning by random search or similar techniques (see e.g. [43]) due to the big computational effort of the simulation. Hyperparameters were thus optimized manually throughout the course of the development of the RL-simulation. We nevertheless point out that the given DQN architecture as well as the used hyperparameters still have room for improvement (see outlook in Section 7).

Table 2 lists all hyperparameters for our DQN implementation. As we use the same terminology as [14] the reader is referred to this paper for more information on the individual parameters.

We implemented the DQN using the tensorflow [44] based Python library Keras [45]. Training was done on a NVIDIA GeForce RTX 2080 Ti.

From input to output, the agent's architecture goes as follows and a graphical representation is given in Fig. 5 (with adaptions of architecture from [14] as described in the section ahead):

- The input consists of an array with the shape 2 $\times$ 2100 $\times$ 2 (see Section 3.3.1).
- One convolutional layer, with 32 filters, a kernel size of 1 $\times$ 16 and a stride of 1 and 8 applying a ReLU activation function [39]
- One convolutional layer, with 64 filters, a kernel size of 1 $\times$ 8 and a stride of 1 and 4 applying a ReLU activation function
- One convolutional layer, with 32 filters, a kernel size of 1 $\times$ 4 and a stride of 1 and 2 applying a ReLU activation function
- One fully connected layer with 256 neurons applying a ReLU activation function
- One fully connected layer with 8 neurons (one per action) as the output layer which applies a linear activation

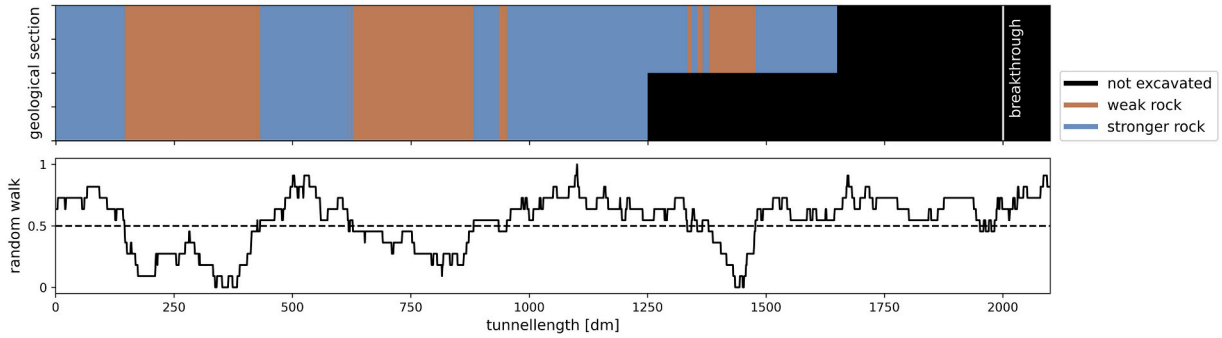

Fig. 4. Top row: an exemplary unique geological section, where brown indicates weak (gt1) and blue stronger rock (gt2). The positions of the top heading and bench are at 165.0 m and 125.0 m respectively. Bottom row: the random walk that is used to generate the geological section. Values above 0.5 are converted to gt2 and below to gt1. Note that the x-axis is the tunnel length in decimeters which corresponds to the number of datapoints of the random walk.

Table 2
Used hyperparameters for the DQN agent. Except for the exploration decay (see Section 3.2.1) all are identical in their meaning as the extended data Table 1 in [14].

Hyperparameter	Value
Replay memory size	100,000
Replay start size	1000
Minibatch size	64
Discount (γ)	0.99
Target network update frequency	10
Initial exploration (ε)	1
Final exploration (ε)	0.05
Learning rate	0.00025
Gradient momentum	0.95
Exploration decay (ε_d)	0.99997

The output of the third convolutional layer is flattened/vectorized before it is fed into the dense layer and we used the mean squared error as a loss function.

3.2.1. Actions

The agent can choose from 8 different actions (Table 3). Possible actions are either top heading – or bench excavation with advance lengths of either 2 or 4 m. In each step the simulation carries out the following operations:

- the agent chooses and executes one of the 8 actions of Table 3 and the respective length of the generated geological section is revealed
- it is calculated/checked if the new position of the excavation face is within a stable area or not (details in Section 3.3.2)

As given in Section 2.1, the support ahead of the face has the effect on the simulation that, there cannot be unstable conditions as long as the excavation face is within the supported area. Because of technical reasons - mostly related to the reward system (see Table 4) - each action is assigned an "action code" (a). An $a < 200$ denotes top heading excavation and $a \geq 200$ bench excavation.

The choice of actions of Table 3 is based on practical engineering experience on the one hand and technical limitations of the RL agent on the other hand. On the practical side, especially the advance length is

Table 3
The eight possible actions the agent can choose from.

Action code (a)	Excavation	Advance length [m]	Face support
110	top heading	2	no
112	top heading	2	yes
150	top heading	4	no
152	top heading	4	yes
200	bench	2	no
202	bench	2	yes
220	bench	4	no
222	bench	4	yes

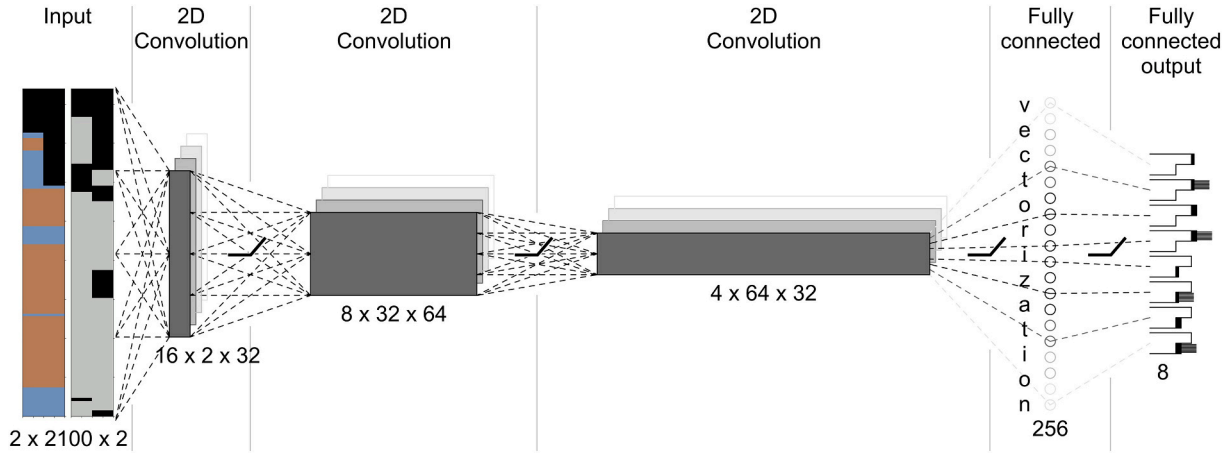

Fig. 5. Schematic representation of the DQN agent's ANN architecture. Note the visualization of rockmass-matrix and the support-matrix to the left. The numbers below each layer are the respective shape of the layer's weights. Dashed connection lines between layers are only for illustrational purposes. Symbols at the output layer represent the eight possible actions (ordered as in Table 3) that are chosen via Q-values by the agent.

highly influential on the stability of the excavation with longer advance lengths being more prone to failures than short ones (e.g. advance lengths in the Austrian standard ÖNORM B2203–1 [46]: 1.0, 1.3, 1.7, 2.2, 3.0 and 4.0 m). After experimenting with advances lengths based on the standards in the beginning, we experienced that a big number of actions leads to a deterioration of the performance/confuses the agent in this model setup. This is also in accordance with other studies which have found that special measures/adaptations of the agent are necessary if the action space becomes increasingly complex [47,48]. The final set of actions in Table 3 therefore aims at giving the agent realistic options to choose from while also keeping the number of actions small (see the outlook in Section 7 for a discussion on increasing the number of actions).

During an episode, the actions are chosen based on an "ε-greedy action selection process", (see also "exploration vs. exploitation trade-off", e.g. [26]). ε is the exploration rate and is initially set to 1 (see "initial exploration in Table 2). Before every move, a number (r) is drawn from a random uniform distribution in the range between 0 and 1. r governs the probability for a move to be a random action (if $r \leq \varepsilon$) or to be based on the agent's prior experience (if $r > \varepsilon$). Throughout training ε will decay following Eq. (2), where the new ε for the next episode (ε_{i+1}) is computed by the previous ε (ε_i) times a constant – the epsilon decay (ε_d).

$$\varepsilon_{i+1} = \varepsilon_i * \varepsilon_d \tag{2}$$

This process guarantees that the agent can explore the environment in the beginning and shifts towards more exploitation of its knowledge towards the end of training. We set the minimum ε to 0.05 below which there is no more decay and ε will be kept constant (see "final exploration" in Table 2).

3.3. Environment

As the DQN agent is the RL pendant to the real life geotechnician, the environment is the RL representation of the rockmass and of logistical processes of the construction site (state) as well as the feedback that is received for better or worse excavation performance (reward). Based on the agent's actions, its main tasks are to update and yield the current state of the construction-site and to provide feedback.

3.3.1. States

The state that is observed by the DQN agent (see Section 3.2) is a hypermatrix of the shape 2 × 2100 × 2 which represents a geological section and a section that shows where support is already installed. The hypermatrix is structured in the following way (see Fig. 6): 2 rows for top heading and bench respectively; 2100 columns for the total length of the tunnel in decimeters plus an additional area beyond the breakthrough (see below); 2 channels for the geological section and the section with the installed support respectively. The values of the hypermatrix – originally ranging from 0 to 2 (see Section 3.1) - are then scaled between 0 and 1. In the channel of the geological section, 0 therefore translates to "not yet excavated area" and 0.5 and 1 represent gt1 and gt2 respectively (see Table 1 and Section 3.1). In the channel of the already installed support, 0 means "no installed support" and 1 means "installed support".

There are 2 terminal states in the simulation (i.e. states that lead to abortion of the simulation after their occurrence):

- A breakthrough is achieved if and only if both pos_{th} and pos_{bi} are $\geq tl$. In this case a breakthrough reward is given (see next section) and the simulation is finished.
- The second terminal state is a timeout which is set to 200 actions; i.e. the agent must achieve a breakthrough within less than 200 actions or otherwise the simulation is aborted, and a negative breakthrough reward is given.

The environment is designed in a way that once pos_{th} or pos_{bi} are $\geq tl$, the position of this part of the excavation is not updated anymore even if it is further excavated. We introduced the "timeout" as we observed in the experimental phase of the study that the agent sometimes reaches a breakthrough with the top heading or the bench but keeps on excavating in the already excavated part of the tunnel. On the one hand this causes the episode to be infinitely long, and on the other hand this leads to excessively large negative penalties which negatively affect the training process.

We set the total length of the sections to 210 m (i.e. tunnellength + maximum possible length of face support) to give the agent the freedom to use actions that install face support even for the last few blasts. From a "real world tunneling" perspective this does not make sense and in the simulation, this would lead to additional penalties, however, the goal is that the agent learns things like this and is not forced to do so.

3.3.2. Reward system

Rewards have the purpose to tell the agent if its actions are beneficial for the total reward. The ultimate goal of the agent is to maximize the return (i.e. sum of the rewards throughout the episode – here: excavate the whole tunnel). During an episode, a reward (also called penalty in case of a negative reward) is given for each individual move that the agent takes [26]. The action is chosen to maximize the expected return of discounted rewards.

Reward systems in RL can vary greatly depending on the given task. While [14], who trained their DQN to play classical Atari 2600 games, took a modified version of the original games' scoring systems as a means to deal with the reward in each timestep, [15] only gave a reward of +1 or − 1 at the end of the game depending on whether or not the agent has won the respective Go-match.

In our tunneling simulation, we designed a hierarchical point system that values (i.e. rewards/penalizes) either a state or an action. The rewarding/penalizing is done based on a list of conditions, that is worked through from top to bottom after every step of an episode and the agent is given the first reward where the condition is fulfilled. Except for the reward for achieving a breakthrough, all rewards are negative (i.

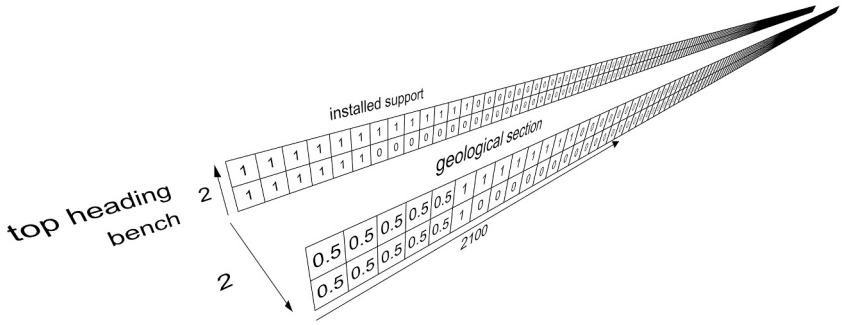

Fig. 6. 3D visualization of the 2 × 2100 × 2 hypermatrix that represents the state of the simulation.

e. penalties) and from top to bottom the penalties are sorted from biggest to smallest. It is therefore possible that multiple conditions are fulfilled in one state, but only the most severe penalty is given. Table 4 shows the list of rewards and their respective conditions.

We designed the list of rewards based on practical engineering experience and requirements arising from training a completely unexperienced agent. For example, we penalize face instabilities higher than a too far distance between top heading and bench excavation, as the immediate safety hazard of an unstable face is bigger. In contrast to that, an even bigger penalty (i.e. -6 points) is triggered if the agent would try to do bench excavation ahead of the top heading excavation. A geotechnician in real world tunneling would know from the start that doing so brings numerous technical-, logistical- and safety problems, but an untrained RL-agent does not.

During the experiments for this study we tried to design the reward system in a way that it is the sum of all penalties that would be fulfilled in a given state and not a hierarchical system. However, doing this was not beneficial for training and seemed to confuse the agent as it apparently did not know what it was punished for. For example, a combination of the penalty for a too far distance between top heading and bench (i.e. -3) plus the penalty for using face support (i.e. -2) would sum up to −5 which is the same penalty as the one for unstable face conditions.

As given in Section 2.1, the evaluation to check if a newly excavated face is stable or not, is done based on the face pressure equation for open face tunneling from [29] (Eq. (3)). In Eq. (3), p_f is the required pressure to achieve stable face conditions, where a $p_f < 0$ indicates stable conditions and a $p_f \geq 0$ indicates unstable conditions, respectively the amount of pressure that is necessary to stabilize the face. γ_R is the ground's unit weight, D the (equivalent) tunnel diameter (see chapter 3.1), d the advance length (describing the unsupported area of the unlined wall), c' the effective cohesion and φ' the effective friction angle.

$$p_f = \gamma_R D * \left(\frac{2 + 3 * \left(\frac{d}{D}\right)^{6 * tan\varphi'}}{18 * tan\varphi'} - 0.05 \right) - \frac{c'}{tan\varphi'} \tag{3}$$

Table 4
Rewards that the agent receives from the environment in response to its actions.

Reward (points)	Description	Condition
tl * 3	reward for achieving breakthrough	$pos_{th} \geq tl$ and $pos_{bi} \geq tl$
tl * 3 * -1	penalty for a timeout	if number of moves in current episode is >200
−6	penalty for using the wrong excavation sequence, i.e. the bench is driven further ahead than the top heading	$pos_{bi} > pos_{th}$
−5	penalty for unstable tunnel face conditions determined from face pressure p_f as evaluated by Eq. (3) (see below)	$p_f \geq 0$
−4	penalty for changing from top heading to bench excavation or vice versa, as this usually involves a delay of the excavation due to logistics	if current $a \geq 200$ and prev. $a < 200$ or if current $a < 200$ and prev. $a \geq 200$
−3	penalty for a too far distance between top heading and bench (see Section 3.1)	if $pos_{th} - pos_{bi} > dist_{max}$ (initially set to 50 m)
−2	penalty for using face support as this consumes additional time and resources	if $a = 112$; $a = 152$; $a = 202$; $a = 222$
−1	penalty for every other move that does not meet any of the above conditions	no other condition is fulfilled

4. Training

In RL, one episode is the whole succession of states in between an initial and a terminal state [26]. In other words, an episode is one whole match of a game, or in this simulation, one whole sequence of actions that ultimately should lead to a breakthrough of the tunnel.

We started training with an ε of 1 (i.e. "initial exploration" in Table 2) to promote exploration in the initial phase of training. The exploration decay of 0.99997 (see Table 2) that decreases ε following Eq. (2) was determined by trial and error. A smaller exploration decay (i.e. faster reaching of the final exploration) has shown to increase instabilities in the training process at an early stage which are presenting themselves in spontaneous increases of the loss and decreases of the reward. With the given exploration decay, the final exploration is reached after 99,858 episodes of training. After this point ε is kept at a constant value of 0.05. Training does not need to be aborted after reaching the final exploration. In the current simulation, we aimed at training the agents for 120,000 episodes to observe one full epsilon decay and some episodes beyond that to check for stable conditions in different rockmasses, as illustrated in Fig. 7 (except for cases where training became unstable at some point; see next section). A copy of the agent is saved after every 1000 episodes.

To observe how big the differences are between individual training runs, we trained several identical DQN agents in the above described environment. After every episode, 21 parameters are saved to monitor the training progress. Below, the parameters that are mentioned in the paper are given (see the code "A_utilities.py" in Appendix 1 for a list of all recorded parameters):

- number of the current episode
- cumulative reward of the episode
- current value of ε
- the number of face instabilities of the whole episode
- average loss of the DQN agent throughout the episode
- number of moves/blasts that were required to finish the episode/to reach a terminal state
- 8 counters for how many times each of the actions of Table 3 were used; the goal of these counters is to see if the agent favors some actions over others and to detect "strategy changes"

5. Experiments

In this section we present five training paths of exemplary agents and discuss the different strategies they found to deal with the given task. In Fig. 7, recordings of the agents' training are given which shows that each agent has found a unique solution, and all training paths are substantially different from one another.

Comparing the five agents to one another, all of them were able to increase the cumulative reward per episode above 200 within 10,000 episodes (Fig. 7 first row). After around 30,000 episodes, first differences arise where first the cumulative reward of agent 2 and then of agent 3 started to stagnate. The reward of agent 1 stagnates at around 80,000 episodes. The cumulative rewards of the agents 4 and 5 kept on increasing, whereas agent 5 became instable after around 75,000 episodes and agent 4 reached the maximum reward at the end of the 120,000 training episodes.

Right from the start, all agents started to use long advance lengths as a means to decrease the number of blasts/moves per episode as this is an effective way to maximize the achievable reward (Fig. 7 second row). Where the agents 1, 4 and 5 all took a similar strategy that aims at continuously minimizing the blasts/moves per episode throughout the whole training, agents 2 and 3 reached a minimum of around 120 blasts/moves per episode after around 30,000 episodes. This correlates well with the stagnating rewards after 30,000 episodes of agents 2 and 3 as described above.

The biggest differences in training paths can be observed with

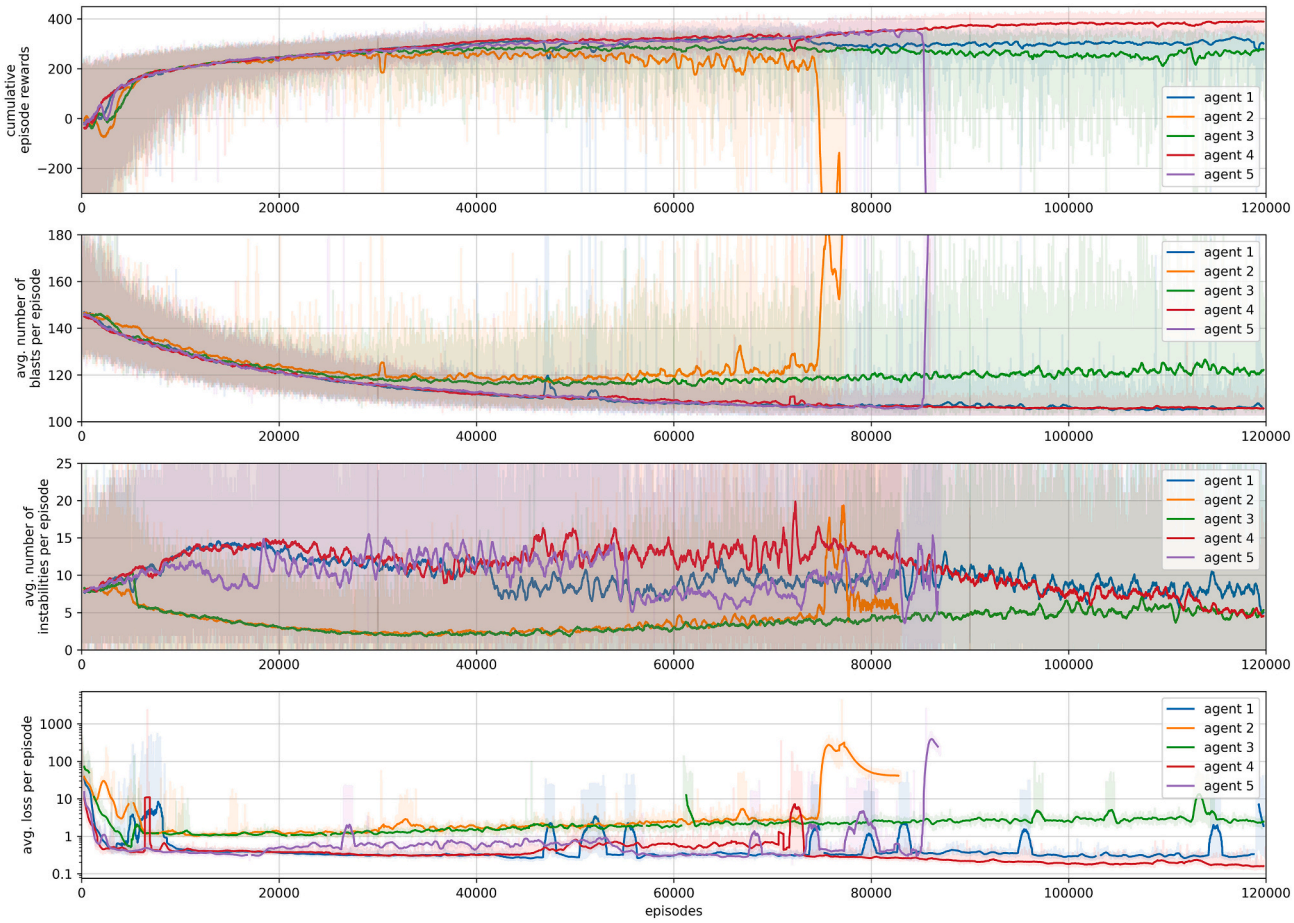

Fig. 7. Different training paths for five exemplary agents over 120,000 episodes. The first row shows cumulative rewards per episode; the second row the average number of blasts/moves that were required to complete each episode; the third row the average number of face instabilities per episode; the fourth row shows the average loss per episode. Transparent colors in the background show the raw data records and the solid lines in the foreground a 500-episode sliding window average.

respect to face instabilities, where only the agents 2 and 3 started to actively decrease the number of face instabilities (Fig. 7 third row) early in the training process (within 10,000 episodes). In contrast to that, the number of face instabilities increased within the first 10,000 episodes for the other agents and then only decreased slowly throughout the rest of the training. The best performing agent 4 shows a remarkable trend of stagnating face instabilities until around 80,000 episodes, followed by a decrease towards the end of training.

While the goal was to let all agents train for 120,000 episodes, the training process of the agents 2 and 5 became unstable after around 75,000 and 85,000 episodes respectively, which led to the abortion of training after it could be observed that the agent would not recover from this. A solution would be to take a version of the agent that was saved before instability occurred and continue training with that, but we left

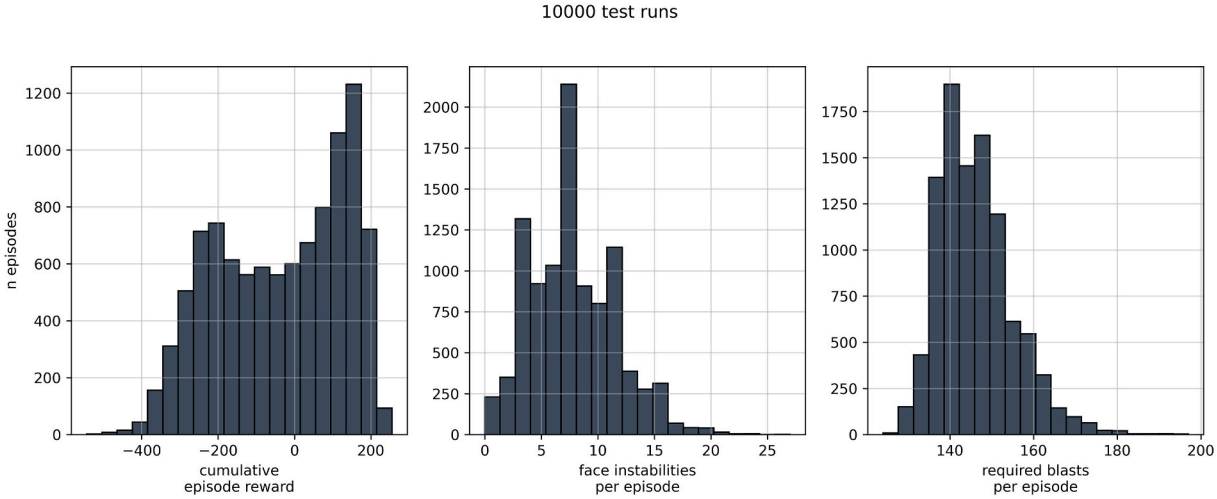

Fig. 8. Histogram for the performance of an agent who plays 10,000 episodes with completely random moves (i.e. $\varepsilon = 1$).

the agents 2 and 5 as they are for explanatory purposes.

Before discussing the performance of the agents, we first establish a baseline for the minimum performance that an agent must reach by letting an agent play with completely random moves (i.e. $\varepsilon = 1$) for 10,000 episodes. In these 10,000 random episodes the agent reached a

- minimum/maximum/median reward of −544/255/-13 points,
- minimum/maximum/median number of 0/27/8 instabilities per episode,
- minimum/maximum/median number of 124/197/145 blasts/moves per episode.

Histograms for the 10,000 random episodes are given in Fig. 8.

The maximum rewards, minimum number of face instabilities and minimum number of blasts/moves that the five agents of Fig. 7 needed are given in Table 5. We computed these numbers based on sliding window averages of 500 episodes as to avoid individual episodes that performed extraordinarily well. Albeit substantial differences can be seen in these statistics, it can generally be observed that all maximum rewards of the agents are above the maximum rewards of the randomly played episodes and therefore training was generally a success.

In these five agents, two groups can be observed, where the agents 2 and 3 reached their maximum performance between episodes 40,000 and 60,000 and the agents 1, 4 and 5 reached their maximum performance between episodes 80,000 and 120,000. Although the first group has reached their peak performance sooner, the achieved reward is generally lower than that of the second group (see description of training paths above).

We conclude this section by presenting the strategy that the best performing agent 4 has found after 119,000 episodes – at its peak performance (in terms of highest reward and low instabilities). We tested the saved agent's checkpoint for 10,000 episodes with a fixed ε of 0.05 which corresponds to the ε at that stage of training and is in accordance to [14] who recommend an $\varepsilon > 0$ also for testing, as this helps the agent to deal with unexpected situations. The histograms of Fig. 9 show the same test statistics as given for the random moves in Fig. 8.

The strategy that the agent adopted is focused on long advance lengths without face support, in alternation with long advance lengths with face support. By doing so the agent avoids face instabilities, while also minimizing the required support (see Table 4 for the respective rewards/penalties). The boxplot of Fig. 10 illustrates this, as it can be observed that the majority of actions is 4 m long advance lengths. Furthermore, the agent focuses on long advance lengths without face support in both excavation types (top heading and bench) which shows that it tries to avoid excessive use of support measures, thus showing a tendency towards economical optimization. It can also be seen that the agent still uses small advance lengths sometimes and, in this case, favors the actions without face support as it has realized that the small advance lengths do not lead to face instabilities in the given conditions.

In Fig. 11 an example of one episode for the agent 4 is visualized. The time-distance diagram in the top row of this figure shows that the agent has learned to optimize the excavation process by minimizing changes between top heading and bench excavation which would be associated with unwanted delays in "real life" tunneling (e.g. building and removing of access ramps). As given in Section 2.1, installed support ahead of the current face always covers 10 m. With the maximum advance length being set to 4 m it would be unnecessary and uneconomical to install face support in two consecutive rounds. In the second row of Fig. 11 it can be seen that within individual sequences of top heading or bench excavation, the agent alternates between supported and unsupported blasts which shows that it has successfully learned to avoid excessive use of support ahead of the face.

6. Conclusion and implications for digital tunneling

Before drawing conclusions from the experiments, it should be pointed out that in this initial study, the agent's possibilities to find creative and not yet imagined solutions to real world tunneling problems are confined to the given set of actions and the taken simplifications (see Section 2.1). We therefore see it as a success that the agent optimizes the given scenario and finds strategies that are comparable to current practices in tunneling. As given in the introduction, the current study should serve as a base for future developments of RL in tunneling. Consequently, optimizations that improve the current practices of "real life" tunneling are to be expected from future studies.

In the experiments of the previous chapter, we can observe that the agents have found policies that minimize the overall amount of necessary blasts, minimize changes between top heading and bench excavation, favor long over short advance lengths and minimize the use of face support. These strategies show that the agents have learned to work in an efficient and economically optimized way. We see similarities in this RL-based tunneling strategy to real world tunneling paradigms like the NATM [49] which uses partial excavation to minimize the necessary support. Minimizing the number of changes between top heading or bench excavation while at the same time not exceeding a too long distance between them is also part of NATM tunneling as this optimizes construction site logistics on the one hand and safety requirements on the other. The found strategies that rely on support ahead of the face show similarities to the "Adeco" method [25] which uses heavy support installation and long advance lengths to deal with the encountered rockmass conditions. While adhering to safety requirements is imperative, most "real life" tunneling methods work towards minimizing the number of necessary blasts which is a policy that was found by all the agents.

TunnRL (see Section 2) has shown that it is not only a functioning environment/simulation of conventional tunneling, but also that a RL agent can successfully interact with it and learn optimized and innovative strategies that seem realistic compared to real world tunnel excavation. At the same time, we see the challenges of computational instabilities and trial and error approach in the process of developing well-functioning models, highlighting this early stage in RL for tunneling. Clearly both the reward system, the rockmass-environment and the action-system has room for improvement. Still we see a significant potential in the TunnRL-concept: firstly, for an on-face decision support system in a further developed and more realistic version, and secondly as a first step to more advanced automation in underground

Table 5
Statistics of the training runs presented in Fig. 7. Values were computed from the 500-episode sliding window average and the episode of the respective value is given in parenthesis behind it. Row-wise best performances are highlighted in grey.

	agent 1	agent 2	agent 3	agent 4	agent 5
max reward	328 (116717)	272 (40586)	294 (62296)	391 (117952)	358 (81411)
min n instabilities	5 (119647)	2 (36173)	2 (34964)	4 (118042)	4 (83402)
min n blasts	106 (104435)	118 (47584)	118 (62180)	106 (105740)	105 (83407)

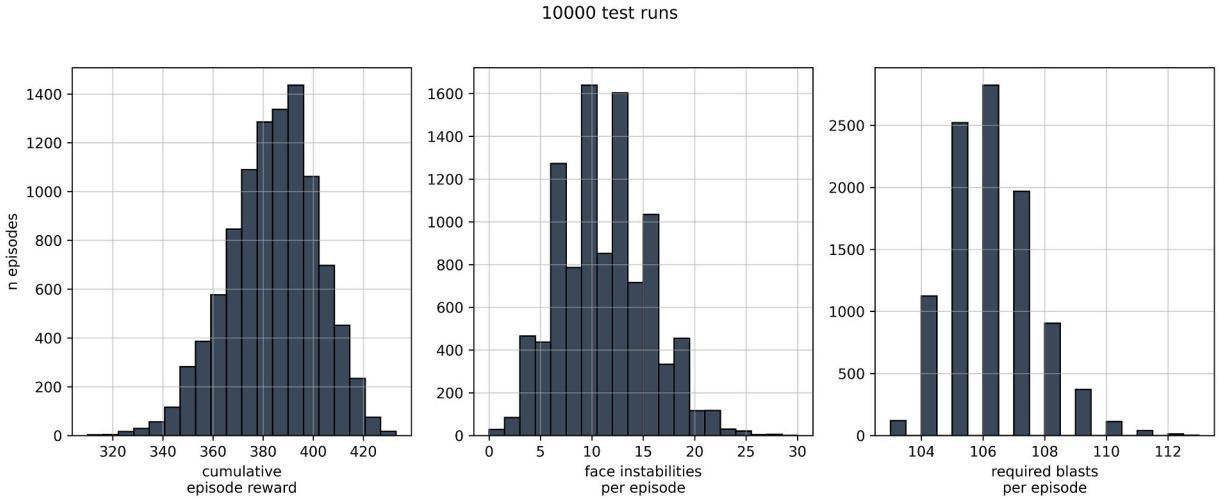

Fig. 9. Histograms showing the performance of the agent 4 that was tested for 10,000 episodes in Fig. 7. Note the different scales of the x-axes in comparison to the histograms of Fig. 8.

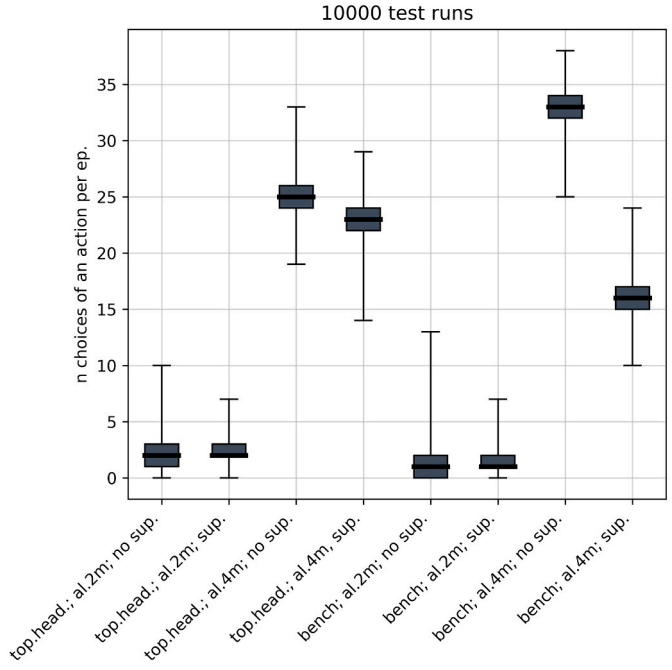

Fig. 10. Boxplot that shows how many times the agent 4 uses each action throughout 10,000 test episodes. Bold black lines in the boxes represent median values; the boxes confine the upper and lower quartiles and the whiskers show min.-max. Values; "al" in the x-labels refers to "advance length".

construction. Where many of the developments of tunnel processes today address the automation of small-scale processes in the tunnel or at the excavation face, a further developed TunnRL could be part of the main controlling mechanism that operates the overall tunnel construction site. Albeit the agents in this simulation were not able to find new and undiscovered strategies of tunnel excavation (see previous chapter), we see the fact that completely untrained agents are able to find tunneling strategies which are comparable to "real world tunneling" as a proof of concept that RL is successfully applicable to this kind of problem. The main goal of the study is therefore fulfilled, and future studies will work towards giving the agent more capabilities and increasing the environment's realism. This will ultimately pave the way for an optimized decision finding process in sequential tunneling.

7. Outlook

Albeit we designed TunnRL in a practice related context, there are numerous improvements to make the framework more realistic and more robust. As the present paper should be the first introduction of RL into tunneling we refrained from over-complicating the simulation and rather give an impetus for future studies that build upon this work. The below given improvements are a non-exhaustive list of ideas that we think are worth to be further explored.

Ever improving processing power will alleviate problems related to computational cost over time. Nevertheless, all improvements must consider that each step in the framework will be done millions of times throughout the training and state of the art RL is by itself already computationally heavy.

Improvements address either the agent or the environment:

- On the environment's side, improvements could work towards making the geotechnical scenario more realistic by involving more ground types and other phenomena like groundwater and in-situ stress conditions. The excavation geometry could become more complex so that also other excavation shapes and sequences such as full-face excavation, or a further division of the top heading and bench excavation are possible.
- Closely related to the excavation geometry – and in our view one of the main points that should be improved – is the stability evaluation of the simulation. The reason why we used Eq. (3) after [29] as a substitute for more sophisticated means of stability assessment (e.g. tunnel cross sectional analyses such as analytical convergence confinement methods or 2D and 3D finite element analysis) is that this analytical solutions is computationally very efficient and does not prolongate the training process too much. For example, 100,000 episodes of the current framework with around 120 moves per episode would require ~1.2e+7 FEM based stability assessments. However, this reduction of the stability assessment is only usable for the ground conditions described in [29] and also neglects other phenomena like the 3D stress state at the tunnel face. Improvements could be to do the stability assessment based on stand-up time concepts that also involve the rockmass quality [50] or ideally 3D finite element analysis as given above.
- Improvements for the agent on the one hand address the agent's performance by modifying the agent's architecture itself. Systematic hyperparameter tuning was not conducted yet and may

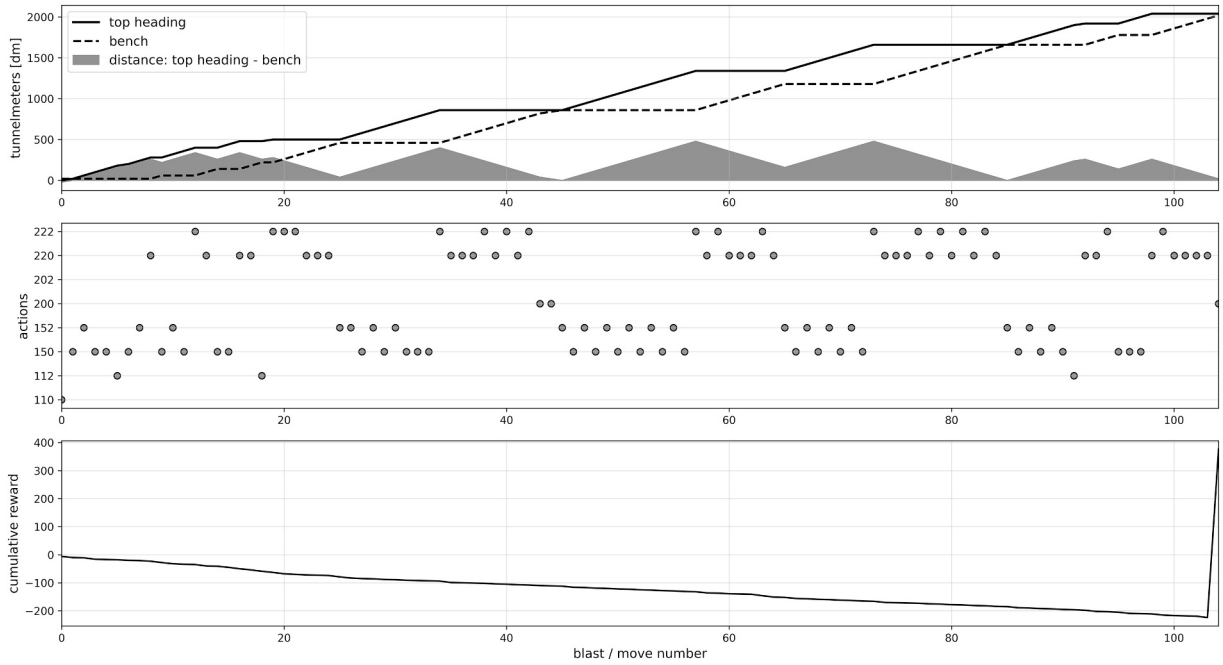

Fig. 11. Exemplary log of one episode. The x-axis shows the number of blasts/moves the agent needed to achieve breakthrough (i.e. 105): Top row: Time-distance diagram showing the logged position of top heading excavation (solid line), bench excavation (dashed line) and the difference in between (grey area). Middle row: Record of the actions (see Table 3 for the corresponding action codes). Bottom row: cumulative reward throughout the episode.

help to further improve the agent's performance (e.g. grid−/random search or even RL based hyperparameter optimization [51]). From a geotechnical point of view though, interesting improvements mainly concern an extension of the agent's capabilities by introducing more possible actions. Whereas direct improvements to the given framework would be more advance lengths and types of tunnel support, other ideas are to involve exploration ahead of the face by simulated measurement while drilling [52] or geophysical exploration [53] to give the agent an idea what might be in front of the current excavation face. An idea in this regard is also to extend the agent to a multi-agent framework as it was used successfully before [15] where different agents have different tasks to fulfill.

Future studies will work towards a more realistic environment and more complex agents in the TunnRL framework. Where TunnRL fits well in the line of the current development of automation, the greatest potential lies in the possibility to develop new and not yet considered tunneling strategies for sequential tunnel excavation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix

Appendix 1: Link to the GitHub repository where this paper's code can be found: https://github.com/geograz/Tunnel-automation-with-Reinforcement-Learning-TunnRL-

References

[1] T. Marcher, G.H. Erharter, M. Winkler, Machine learning in tunnelling – capabilities and challenges, Geomechanik Tunnelbau 13 (2020) 191–198, https://doi.org/10.1002/geot.202000001.

[2] DAUB, BIM in Tunnelling: Digital Design, Building and Operation of Underground Structures. http://www.daub-ita.de/fileadmin/documents/daub/gtcrec4/gtcrec11v3_BIM_in_Tunnelling_05-2019.pdf, 2019 (accessed 9 October 2020).

[3] L. Wagner, A. Kluckner, C.M. Monsberger, P. Wolf, K. Prall, W. Schubert, W. Lienhart, Direct and distributed strain measurements inside a Shotcrete lining: concept and realisation, Rock Mech. Rock. Eng. 53 (2020) 641–652, https://doi.org/10.1007/s00603-019-01923-4.

[4] G.H. Erharter, T. Marcher, C. Reinhold, Application of artificial neural networks for underground construction – chances and challenges – insights from the BBT exploratory tunnel Ahrental Pfons, Geomechanik Tunnelbau 12 (2019) 472–477, https://doi.org/10.1002/geot.201900027.

[5] G.H. Erharter, T. Marcher, C. Reinhold, Comparison of artificial neural networks for TBM data classification, in: Proceedings of the 14th International Congress on Rock Mechanics and Rock Engineering (ISRM 2019), Foz de Iguassu, Brazil, 2019.

[6] G.H. Erharter, T. Marcher, C. Reinhold, Artificial neural network based online rockmass behavior classification of TBM data, in: Information Technology in Geo-Engineering, firstst ed. twentiethtwentieth, Springer, 2020, pp. 178–188.

[7] R. Wu, Y. Fujita, K. Soga, Integrating domain knowledge with deep learning models: an interpretable AI system for automatic work progress identification of NATM tunnels, Tunn. Undergr. Space Technol. 105 (2020) 103558, https://doi.org/10.1016/j.tust.2020.103558.

[8] S.K. Shreyas, A. Dey, Application of soft computing techniques in tunnelling and underground excavations: state of the art and future prospects, Innov. Infrastruct. Solut. 4 (2019), https://doi.org/10.1007/s41062-019-0234-z.

[9] W. Zhang, R. Zhang, C. Wu, A.T.C. Goh, S. Lacasse, Z. Liu, H. Liu, State-of-the-art review of soft computing applications in underground excavations, Geosci. Front. 11 (2020) 1095–1106, https://doi.org/10.1016/j.gsf.2019.12.003.

[10] S. Isam, Z. Wengang, Use of soft computing techniques for tunneling optimization of tunnel boring machines, Underground Space (2020), https://doi.org/10.1016/j.undsp.2019.12.001.

[11] B.B. Sheil, S.K. Suryasentana, M.A. Mooney, H. Zhu, Machine learning to inform tunnelling operations: recent advances and future trends, Proc. Inst. Civ. Eng. (2020) 1–18, https://doi.org/10.1680/jsmic.20.00011.

[12] Q. Zhang, Z. Liu, J. Tan, Prediction of geological conditions for a tunnel boring machine using big operational data, Autom. Constr. 100 (2019) 73–83, https://doi.org/10.1016/j.autcon.2018.12.022.

[13] G.H. Erharter, T. Marcher, MSAC: towards data driven system behavior classification for TBM tunneling, Tunn. Undergr. Space Technol. 103 (2020) 103466, https://doi.org/10.1016/j.tust.2020.103466.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (2015) 529–533, https://doi.org/10.1038/nature14236.

[15] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of go with deep

neural networks and tree search, Nature 529 (2016) 484–489, https://doi.org/10.1038/nature16961.
[16] J. Shahrabi, M.A. Adibi, M. Mahootchi, A reinforcement learning approach to parameter estimation in dynamic job shop scheduling, Comput. Ind. Eng. 110 (2017) 75–82, https://doi.org/10.1016/j.cie.2017.05.026.
[17] M. Lanham, Reinforcement Learning, from Games to Geologic Interpretation. https://medium.com/@cxbxmxcx/reinforcement-learning-from-games-to-geologic-interpretation-93757664f0e4, 2019 (accessed 2 October 2020).
[18] M. Lanham, Reinforcement Learning, It's Coming not Just for Games. https://medium.com/@cxbxmxcx/reinforcement-learning-its-coming-not-just-for-games-6b6064e02bbe, 2019.
[19] D. Schwung, A. Schwung, S. Ding, Actor-critic reinforcement learning for energy optimization in hybrid production environments, Int. J. Comput. 18 (2019) 360–371.
[20] I. Carlucho, M. de Paula, S. Wang, Y. Petillot, G.G. Acosta, Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning, Robot. Auton. Syst. 107 (2018) 71–86, https://doi.org/10.1016/j.robot.2018.05.016.
[21] B. Maidl, M. Thewes, U. Maidl, D. Sturge, Handbook of Tunnel Engineering, First, Engl. ed., Ernst/Wiley, Berlin, 2013.
[22] W. Stipek, R. Galler, M. Bauer (Eds.), 50 Years of NATM: Experience Reports, ITA, Austria, Wien, 2012.
[23] Norwegian Tunnelling Society (Ed.), Norwegian Tunnelling Technology: Publication no. 23, 2014.
[24] S. Pelizza, D. Peila, Soil and rock reinforcements in tunnelling, Tunn. Undergr. Space Technol. 8 (1993) 357–372, https://doi.org/10.1016/0886-7798(93)90020-V.
[25] P. Lunardi, Design and Construction of Tunnels: Analysis of Controlled Deformation in Rocks and Soils (ADECO-RS), Springer-Verlag, Berlin, Heidelberg, 2008.
[26] S. Raschka, V. Mirjalili, Python Machine Learning - Third Edition: Machine learning and deep learning with python, scikit … -learn, and tensorflow 2, Packt Publishing Limited, [S.l.], 2019.
[27] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, Second edition, The MIT Press, Cambridge Massachusetts, 2018.
[28] ITA Working Group Conventional Tunnelling, General Report on Conventional Tunnelling Method, Distrib, ITA Secretariat c/o EPFL, Lausanne, 2009.
[29] P.A. Vermeer, N. Ruse, T. Marcher, Tunnel heading stability in drained ground, Felsbau 20 (2002) 8–18.
[30] M.A. Kanji, Critical issues in soft rocks, J. Rock Mech. Geotech. Eng. 6 (2014) 186–195, https://doi.org/10.1016/j.jrmge.2014.04.002.
[31] T. Marcher, S. Stauder, M. Winkler, HSSR – Ein Versuch der Einordnung und Abgrenzung des Materials, in: T. Marcher (Ed.), Beiträge zum 1. Hard Soil – Soft Rock (HSSR) Minisymposium: Charakterisierung, Modellierung und experimentelle Untersuchungen von Uebergangsgesteinen – Aktuelles aus Forschung und Entwicklung, Graz, 2020.
[32] G. Anagnostou, K. Kovári, Face stability conditions with earth-pressure-balanced shields, Tunn. Undergr. Space Technol. 11 (1996) 165–173, https://doi.org/10.1016/0886-7798(96)00017-X.
[33] A.A. Borovkov, Random walks and factorisation identities, in: A.A. Borovkov (Ed.), Probability Theory, Springer London, London, 2013.
[34] C.J.C.H. Watkins, Learning from Delayed Rewards. Ph.D. Thesis, Oxford, 1989.
[35] C.J.C.H. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (1992) 279–292, https://doi.org/10.1007/BF00992698.
[36] A.A. Javadi, M. Rezania, Applications of artificial intelligence and data mining techniques in soil modeling, Geomech. Eng. 1 (2009) 53–74, https://doi.org/10.12989/gae.2009.1.1.053.
[37] M.A. Shahin, M.B. Jaksa, H.R. Maier, Recent advances and future challenges for artificial neural Systems in Geotechnical Engineering Applications, Adv. Artific. Neural Syst. 2009 (2009) 1–9, https://doi.org/10.1155/2009/308239.
[38] H. Kinsley, Reinforcement Learning W/ Python. https://pythonprogramming.net/q-learning-reinforcement-learning-python-tutorial/, 2019.
[39] R.H.R. Hahnloser, R. Sarpeshkar, M.A. Mahowald, R.J. Douglas, H.S. Seung, Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit, Nature 405 (2000) 947–951, https://doi.org/10.1038/35016072.
[40] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: G. Gordon, D. Dunson, M. Dudik (Eds.), Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.
[41] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (1998) 2278–2324, https://doi.org/10.1109/5.726791.
[42] D.H. HUBEL, T.N. WIESEL, Shape and arrangement of columns in cat's striate cortex, J. Physiol. 165 (1963) 559–568, https://doi.org/10.1113/jphysiol.1963.sp007079.
[43] F. Chollet (Ed.), Deep Learning with Python, Manning, Shelter Island, NY, 2018.
[44] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/, 2015.
[45] F. Chollet, Others, Keras, 2015.
[46] Österreichisches Normungsinstitut, Untertagebauarbeiten: Teil 1: Zyklischer Vortrieb, Wien 91.010.20; 93.020, 2019.
[47] Z. Zhao, Y. Liang, X. Jin, Handling large-scale action space in deep Q network, in: 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD), IEEE, Chengdu, 2018, pp. 93–96.
[48] T. Zahavy, M. Haroush, N. Merlis, D.J. Mankowitz, S. Mannor, Learn what not to learn: action elimination with deep reinforcement learning, in: Advances in Neural Information Processing Systems 31 (NIPS 2018), Montreal, Canada, 2018, pp. 3566–3577.
[49] ÖGG, Guideline for the Geotechnical Design of Underground Structures with Conventional Excavation: Ground characterization and coherent procedure for the determination of excavation and support during design and construction. Translated from version 2.1, second.first, Salzburg, 2010.
[50] H. Lauffer, Gebirgsklassifizierung für den Stollenbau, Geol. Bauwesen 24 (1958) 46–51.
[51] P. Zhang, H. Li, Q.P. Ha, Z.-Y. Yin, R.-P. Chen, Reinforcement learning based optimizer for improvement of predicting tunneling-induced ground responses, Adv. Eng. Inform. 45 (2020) 101097, https://doi.org/10.1016/j.aei.2020.101097.
[52] J. van Eldert, H. Schunnesson, D. Johansson, D. Saiang, Application of measurement while drilling technology to predict rock mass quality and rock support for Tunnelling, Rock Mech. Rock. Eng. 53 (2020) 1349–1358, https://doi.org/10.1007/s00603-019-01979-2.
[53] A. Radinger, F. Fasching, G. Pack, I. Kreutzer, D. Kostial, Consistent exploration by probe drilling and TSWD through the example of the Koralm tunnel/Konsequente Vorauserkundung mittels Bohrungen und TSWD am Beispiel des Koralmtunnels, Geomechanik Tunnelbau 7 (2014) 540–550, https://doi.org/10.1002/geot.201400038.