Neural Networks 144 (2021) 699–725

Contents lists available at ScienceDirect

Neural Networks

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

2021 Special Issue on AI and Brain Science

Decentralized control and local information for robust and adaptive decentralized Deep Reinforcement Learning☆

Malte Schilling [a,*], Andrew Melnik [b], Frank W. Ohl [c,d], Helge J. Ritter [b], Barbara Hammer [a]

[a] Machine Learning Group, Bielefeld University, 33501 Bielefeld, Germany
[b] Neuroinformatics Group, Bielefeld University, 33501 Bielefeld, Germany
[c] Department of Systems Physiology of Learning, Leibniz Institute for Neurobiology, Magdeburg, Germany
[d] Institute of Biology, Otto-von-Guericke University, Magdeburg, Germany

ARTICLE INFO

Article history:
Available online 1 October 2021

Keywords:
Deep Reinforcement Learning
Motor control
Decentralization
Local information

ABSTRACT

Decentralization is a central characteristic of biological motor control that allows for fast responses relying on local sensory information. In contrast, the current trend of Deep Reinforcement Learning (DRL) based approaches to motor control follows a centralized paradigm using a single, holistic controller that has to untangle the whole input information space. This motivates to ask whether decentralization as seen in biological control architectures might also be beneficial for embodied sensori-motor control systems when using DRL. To answer this question, we provide an analysis and comparison of eight control architectures for adaptive locomotion that were derived for a four-legged agent, but with their *degree of decentralization* varying systematically between the extremes of fully centralized and fully decentralized. Our comparison shows that learning speed is significantly enhanced in distributed architectures—while still reaching the same high performance level of centralized architectures—due to smaller search spaces and *local costs* providing more focused information for learning. Second, we find an *increased robustness of the learning process* in the decentralized cases—it is less demanding to hyperparameter selection and less prone to becoming trapped in poor local minima. Finally, when examining *generalization to uneven terrains*—not used during training—we find best performance for an intermediate architecture that is decentralized, but integrates only *local information* from both neighboring legs. Together, these findings demonstrate beneficial effects of distributing control into decentralized units and relying on local information. This appears as a promising approach towards more robust DRL and better generalization towards adaptive behavior.

1. Introduction

Adaptive behavior in animals is characterized by an extraordinary robustness with respect to facing highly varying and unpredictable environmental conditions (Beer, 1990; Beer, Chiel, & Sterling, 1990). Consider, for example, an insect that is able to climb through a twig even though there is only very limited information on possible footholds, and it is impossible to predict the movements of the substrate. Such traits of adaptivity have motivated attempts to equip technical systems with similar performance, and several different strategies have been followed over the years. For our purpose, we will concentrate on two: First, biologically-inspired control approaches focus on biological control principles and transfer these into well designed settings and experimental setups. This often leads to carefully hand-crafted control structures that are well explainable and test specific hypotheses on detailed mechanisms of control (Ijspeert, 2014; Schilling, Hoinville, Schmitz and Cruse, 2013). But this involves fine tuning of the system and its parameters which limits generalization to broader contexts and application in natural environments. Second, in learning-based approaches, the main focus is typically on finding fitting parameters in an automatic fashion that optimize a given objective (Billard & Kragic, 2019; Cully, Clune, Tarapore, & Mouret, 2015). In particular, Deep Reinforcement Learning (DRL) has shown to be successful over the last years (Hwangbo et al., 2019; Kidziński et al., 2018). It has demonstrated to produce well performing control policies for application in robots in a range of environments. However, even the most current learning-based systems still have difficulties when facing noisy environmental settings, and they do not

☆ This research was supported by the research training group "DataNinja" (Trustworthy AI for Seamless Problem Solving: Next Generation Intelligence Joins Robust Data Analysis) funded by the German federal state of North Rhine-Westphalia and by the Cluster of Excellence Cognitive Interaction Technology CITEC (EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG).

* Corresponding author.
E-mail address: mschilli@techfak.uni-bielefeld.de (M. Schilling).

https://doi.org/10.1016/j.neunet.2021.09.017

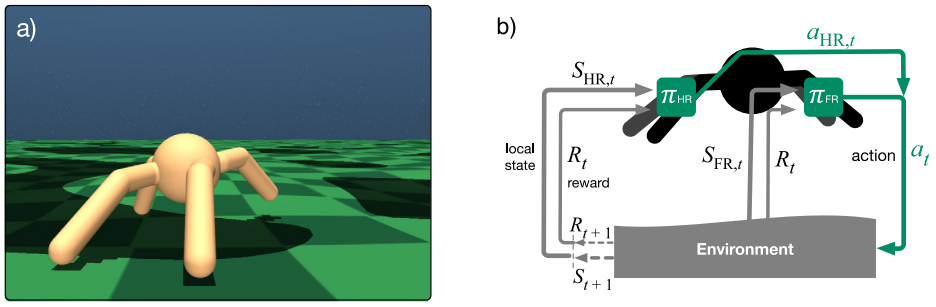

Fig. 1. Visualization of our decentralized approach for locomotion of a simulated robot. In (a) the modified quadruped walking robot is shown (derived from an OpenAI standard DRL task, but using more realistic dynamic parameters and a modified reward function). In (b) a sketch of a (fully) decentralized approach is shown: On the one hand, control is handled concurrently and there are multiple controller (only two are shown in green in the visualization), one for each leg which reduces the action space of each individual controller (e.g., $a_{HR,t}$; FL — front left leg, HL — hind left leg, HR — hind right leg, FR — front right leg). On the other hand, only limited input information is used as a state (gray arrows on the left, $S_{HR,t}$ and $S_{FR,t}$), in this case only information from that particular leg which dramatically reduces the input state space. Control policies are trained using DRL which is driven by a reward signal (R_t, as a simplification this is shown as shared between all controller). This overall simplifies learning of robust behavior considerably and leads, as will be shown, to better generalization.

perform well when facing slight changes in the appearance or configuration of an environment. It appears as a problem that these systems aim to solve one—or even multiple—specific tasks in a static context (for review see Neftci & Averbeck, 2019; or see Finn, Abbeel, & Levine, 2017 highlighting problems when quick adaptations are required), and that these learning-based systems still show a tendency of overfitting (Zhang, Vinyals, Munos, & Bengio, 2018). Therefore, there is today a growing interest in bringing the two kinds of approaches together: Guide learning of optimized-oriented control approaches through insights from neurobiology (Hassabis, Kumaran, Summerfield, & Botvinick, 2017) which aims at facilitating learning and improving robustness of control. Following this rationale, this article aims at a biologically inspired and learning-based approach that is applied to control of locomotion of an animal-like simulated walking robot.

One key insight for the organization of motor control in animals is modularity (Clune, Mouret, & Lipson, 2013; Dickinson et al., 2000) which can be understood as an organization in which a network consists of *"multiple densely connected clusters, each with only a limited connection to other clusters"* (Ellefsen, Huizinga, & Torresen, 2020, p. 3). We will distinguish two main characteristics of modularity of motor control as found in animals: On the one hand, hierarchical organization—actions can be decomposed into sub-actions on different levels of a hierarchy which allows for flexible recombination and induces temporal abstraction (Binder, Hirokawa, & Windhorst, 2009; Haruno, Wolpert, & Kawato, 2003; Mengistu, Huizinga, Mouret, & Clune, 2016; Uithol, van Rooij, Bekkering, & Haselager, 2012). On the other hand—and as a key characteristic that will set our learning approach apart—, we focus on the modular and decentralized structure of the motor control system.

A hierarchical structure is decomposing complexity into different modules and leading to a form of vertical (as well as temporal) abstraction in which higher levels activate modules on the lower level, e.g., in locomotion a higher level routine for walking at a certain speed would sequentially activate low-level motor primitives for swing and stance movements. Such hierarchical approaches have been widely applied in motor control (Binder et al., 2009; Dickinson et al., 2000) and in hierarchical forms of DRL (Merel, Botvinick, & Wayne, 2019). Hierarchical approaches are beneficial as they can apparently deal with a wider variety of contexts. But, mostly this success is a result of transfer learning and the ability to switch between different contexts which allows exploiting a control structure of a similar context. This has shown to work well in examples where different low-level motor primitives were used, for example, to control walking behaviors that are combined on a higher level for navigation. Such approaches also tend to work well when dealing with severe intervention, for example, loss of a leg (Schilling, Ritter, & Ohl, 2019). But this still differs from the range of exquisite adaptivity found in animals. In the case of the above mentioned insect, climbing through a twig, behavior is continuously adapted to an unpredictable environment as it is sensed in real-time. This appears to require other mechanisms as well that act on a much faster timescale.

Along this line, we argue that decentralization (Clune et al., 2013; Ellefsen et al., 2020) is a further important characteristic of motor control found in biology. Decentralization describes the general idea of a factorization of a system into concurrent local modules—distributed over the whole nervous system—that act concurrently on local information and that allow for fast, local computations realizing, e.g., reflex-pathways (Clune et al., 2013) as found in animals and humans (Alon, 2006; Mountcastle, 1997). This local information is directly factorized based on the availability of only nearby sensory inputs. Such decentralized control modules are partially autonomous as they are driven by local signals, but could be modulated by higher levels. Overall, adaptive behavior emerges from the interaction of the decentralized control mechanisms. Decentralization can be understood as complementing modularity along a hierarchy which is focussing on abstraction along a vertical axis. It describes the general idea of concurrent, autonomous modules—potentially on the same level of the hierarchy—that allow for fast, local computations. Such a decentralized control structure appears to be crucial and beneficial for adaptivity of walking, and it is well described in insects (Bidaye, Bockemühl, & Büschges, 2018; Dürr, Schmitz, & Cruse, 2004) which we will take as direct inspiration and which is missing from current—Deep Reinforcement and other—optimization-oriented approaches. In biology, it is often reasoned that decentralization is a necessity because of slow transmission of information in sensory pathways that requires computation closer to the sensors. As we do not consider delays along information pathways, we take a different perspective in which we aim to understand, if decentralization as such provides advantages during learning and for robust control.

We propose a decentralized architecture for motor control in which control is not handled by one single, holistic control unit. Instead, control is distributed onto multiple local control modules which run concurrently (for a simple sketch illustrating the idea, see Fig. 1). Such decentralized control structures can be much simpler as they only rely on local sensory information of a lower dimensionality which should benefit learning. We demonstrate how a decentralized control structure can learn to walk for a four-legged simulated robot using DRL. Our goal is to analyze

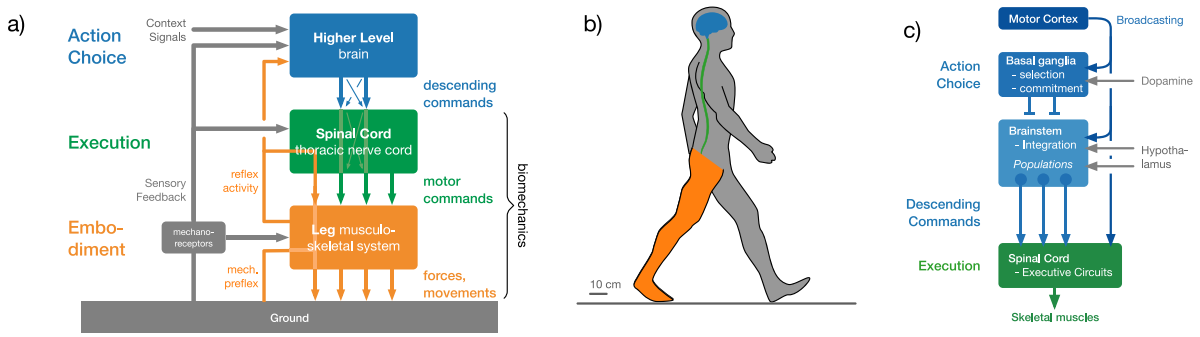

Fig. 2. Hierarchical organization in motor control: Panel (a) shows a schematic of hierarchical organization of motor control. Importantly—providing more detail compared to typical textbook illustrations (Magill & Anderson, 2017)—the different levels are interconnected through highly parallel connections. This highlights the concurrent and decentralized structure of the system. Colors represent different levels of the motor hierarchy: A higher level is shown in blue, an intermediate (and decentralized) control level is shown in green. Interactions with the environment (including preflexes and properties of muscles) are shown in orange (color coding of different control levels applies to Figs. 1, 2, 3, and 4). Such an organization is found in both, humans and animals (Dickinson et al., 2000). Panels (b) and (c) illustrate this for the example of humans: (b) gives an overview of embodied control in humans. Panel (c) shows a schematic for hierarchical motor control and circuits for body movements in humans (adapted from Arber & Costa, 2018): Movements require coordinated activation of neuronal populations across different parts of the nervous system. On a high level (blue), an action is selected. High level centers project onto sensorimotor cortex that further broadcasts to basal nuclei, brain stem, and spinal cord. These circuits transmit information concurrently to brainstem command lines (Graziano, 2006). Descending command lines activate executive circuits in the central nervous system (shown in green). On the lowest level (shown in orange in (a) and (b)), executive circuits control actuators that govern body movements. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and demonstrate the benefits of this decentralized and modular control structure in changing and challenging environments. As our previous work has already shown—in a proof of concept—that local information can be sufficient to learn usable walking controllers, this raises the questions of (1) what kind of structure and information is required for control of locomotion, (2) how does this affect learning, and, in particular, (3) how does this affect generalization performance in unseen environments. In this article, we will, first, address the influence of, on the one hand, differing structures of control, and, on the other hand, local and more distal information. Towards this goal, we will use variations of control structures for the four-legged simulated robot. Even though our approach is inspired by work on insects, we are choosing a four-legged robot as this allows to vary the inputs systematically from a purely fully decentralized control approach based solely on local information—in which case the control of a leg is only based on sensory information from that particular leg—up to the current standard paradigm in DRL in which there is one central controller that has access to all sensory information. This will, on the one hand, confirm earlier results demonstrating that local structures are sufficient (Schilling, Konen, Ohl, & Korthals, 2020). But furthermore, the four-legged robot allows for a detailed analysis in varying environments which will point out that some form of coordination and information exchange between legs is advantageous when facing more challenging, uneven terrain. Local controllers that integrate some information of neighboring legs will demonstrate more robust behavior with respect to change of environment and, surprisingly, much more robust towards selection of hyperparameters for learning which overcomes one of the typical shortcomings of DRL. The article is structured as follows: In the second section, an overview on insights from biology on the structure of motor control systems will be given. Section three will present a broad overview of related work, focussing, on the one hand, on more biologically-inspired approaches and, on the other hand, on learning based-approaches. Section four will introduce the different control architectures we use in our Deep Reinforcement Learning experiments and will explain how these differ with respect to decentralization, locality of information, and structure of reward. In the result section, five larger experiments are presented, and the article concludes in a discussion and summary.

2. Insights from biological control of locomotion: Hierarchical organization and decentralization

Adaptive behavior allows animals to produce stable behavior across a variety of different contexts. They flexibly adapt to changing specific environmental conditions and can deal with variations under these conditions even when facing uncertainty. This robustness to uncertainty still sets animal behavior apart from current engineering and learning solutions. The following section will summarize general insights on the organization of motor control as found in animals. In particular, our focus will be on hierarchical organization and decentralization.

2.1. Hierarchical organization in biological motor control

The difficult task of controlling a complex system is addressed in many animals (including insects) through hierarchical organization and modularization of the control system in which the complexity is distributed onto different levels of a motor hierarchy (Botvinick, 2008; d'Avella, Giese, Ivanenko, Schack, & Flash, 2015) and split into functional modules (Alon, 2006) (see Fig. 2). A highest level deals with selecting goal-directed behaviors. On an intermediate level, actions are selected (Arber & Costa, 2018) depending on context (Fig. 2(c)). This leads to an internal competition between different actions that is context dependent. A lower level realizes motor control primitives (or synergies) (Giszter, Mussa-Ivaldi, & Bizzi, 1993; Hart & Giszter, 2010) that are modulated by the higher levels through descending commands (Fig. 2(c), lower level shown in green with the higher levels' projections shown as descending commands).

At the lower level, motor primitives allow for fast, sensory-guided adaptation towards disturbances. These describe how different muscles are working in concert for performing a specific action. Importantly, as a result of this distribution of complexity, the lower level is focused on small groups of muscles which leads to a decentralized organization and concurrent operation of multiple such lower level motor primitives. Such a modularization can be found in vertebrates and invertebrates (Flash & Hochner, 2005; Pearson, 1995). In vertebrates and higher animals the research focus is usually on the higher levels (Fig. 2(c) provides a good summary of such a state-of-the-art view Arber & Costa, 2018, and similar ones can be found, e.g., in textbooks as Magill & Anderson, 2017; higher levels shown in blue). Such a high level

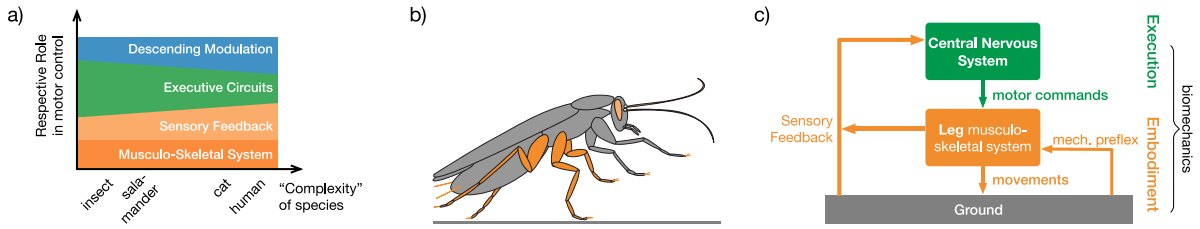

Fig. 3. Contributions to hierarchical motor control: A hierarchical organization of motor control is known as a general control strategy for vertebrate locomotion (Grillner, 2003) and the overall structure is assumed for other animals as well (Dickinson et al., 2000). This is visualized in (a) in a schematic of differential roles of four components that underly locomotion across animals (following and adapted from Ijspeert, 2018; this is not meant as a quantitative characterization, but should point out shared characteristics between animal species in which complexity is only meant as a rough ordering of animal species). Research on vertebrates and humans (see Fig. 2(b) and (c)) puts a focus on higher level processing in the brain and descending commands which is nicely complemented by research on embodiment and the contribution of mechanical properties in insects shown in (b) and (c) (adapted from Dickinson et al., 2000). Panel (b) shows a fast running cockroach highlighting sensors and actuators that are in interaction with the environment (in orange). In (c) a general low level scheme is given (complementing our general scheme given in Fig. 2(a)), highlighting that neural and mechanical feedback play important roles in control of locomotion: On an intermediate level (green), the central nervous system produces motor commands that activate the musculo-skeletal system of the animal which leads to movement. Sensory input from multiple modalities is routed back to the central nervous system and modulates motor commands. In parallel, mechanical preflexes directly act to resist perturbations. While this is visualized for insects here, it represents a general model for locomotor control, and such structures are shared with other invertebrates and mammals (for more details see Dickinson et al., 2000). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

view is nicely complemented by work in invertebrates (Fig. 3) that demonstrates, on a detailed level, the importance of the lower-level motor primitives as these are modulated by sensory inputs and descending commands from higher levels (shown in green in Figs. 2 and 3). It further stresses the contribution of mechanical properties and embodiment (Chiel & Beer, 1997; Chiel, Ting, Ekeberg, & Hartmann, 2009; Nishikawa et al., 2007) in a control task as locomotion (shown in orange in Figs. 2(a) and 3).

One well-studied example is legged locomotion in insects which spans a behavioral continuum from slow walking to fast running. During fast running, the effectiveness of sensory feedback would be constrained by sensorimotor delays (More & Donelan, 2018) as, for example, at top running speed, a cockroach lifts each of its six legs 20 times per second, corresponding to a step period of only 50 ms (Full & Tu, 1991). This might not leave enough time for sensory feedback to adjust leg movements on a step-by-step basis (Jindrich & Full, 2002; Zill & Moran, 1981). Therefore, it is assumed that fast running in insects is driven predominantly by central oscillating units (Fig. 3(c) shown in green) in a feed-forward fashion (Bidaye et al., 2018). A shift towards more feedforward control in (fast) locomotion is assumed in other animals as well (Clancy, Orsolic, & Mrsic-Flogel, 2019). Importantly, insects can still recover from perturbations, such as uneven terrain, during fast running (Jindrich & Full, 2002; Sponberg & Full, 2008). This is enabled by passive forces from the musculo-skeletal system (Ache & Matheson, 2013; Dudek & Full, 2006), which act more quickly than sensory reflexes as mechanical "preflexes" (Brown & Loeb, 2000) (shown in orange in Fig. 3(b) and (c)). Dickinson et al. (2000) pointed out that such properties of embodiment are an important part of adaptive behavior and interacting with an environment for all animals (Fig. 3(c) shows a sketch of his conceptualization and (a) adds an overview by Ijspeert (2018) on contributions of different factors across a spectrum of animals). Exploiting mechanical properties of the body (e.g., muscles) and mechanical preflexes can facilitate fast running and can compensate for small disturbances (for another example see McGeer, 1993, passive walkers).

2.2. Decentralization in biological motor control

Top-down driven and feedforward processing is assumed to drive control of fast walking and running. But, on the other side of a spectrum of different walking velocities, during slow walking insects place their legs accurately in space, driven by detailed sensory information about the body and the environment (Niven, Ott, & Rogers, 2012; Theunissen, Vikram, & Dürr, 2014). This

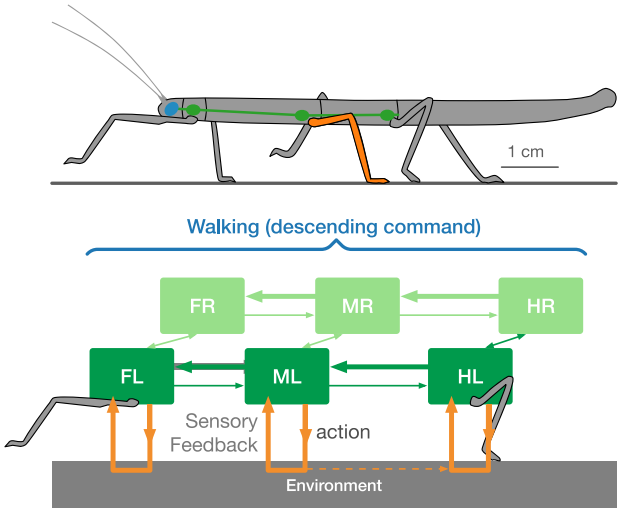

Fig. 4. Decentralized motor control structure: On top, schematic of a stick insect is shown. Bottom, schematic of decentralized organization of the stick insect control system which has been used as a model for six-legged robots (FL — front left leg, ML — middle left leg, HL — hind left leg, FR — front right leg, MR — middle right leg, HR — hind right leg). Colors correspond to the different levels of the motor hierarchy: Higher level is, again, shown in blue (e.g., information on walking direction). Local leg control level is shown in green (Front, Middle, Hind leg on Left and Right side) with coordination influences between neighboring legs shown as arrows. Interaction with the environment (including preflexes and properties of muscles) are shown in orange. Behavior emerges as a result of decentralized and locally interacting concurrent control structures (Schilling, Hoinville et al., 2013). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

is critical when, for example, traversing cluttered environments such as canopies, in which secure footholds are sparse. Behavioral results from insect walking studies show here a wide diversity of walking behavior and not just a small number of fixed gait patterns (Bidaye et al., 2018; DeAngelis, Zavatone-Veth, & Clark, 2019). Temporal coordination of locomotion appears better characterized as free gaits in which temporal relations emerge from the interaction with the environment. This allows to constantly adapt locomotion to unpredictable environments and to adjust the temporal coordination as required. It further has shown to be energy efficient (Nishii, 2006). A decentralized control structure appears to be crucial and beneficial for adaptivity of walking. Such an organizational structure of motor control in insects is well described (Bidaye et al., 2018; Dürr et al., 2004). On the one

hand, this agrees with a hierarchical organization, as there is a higher level producing decisions which behaviors to perform. On the other hand, these findings further point out that biological control acts in a concurrent and modular fashion. Motor control for walking in insects is assumed to be constituted of local control modules. There exists one individual controller for each leg that switches local behaviors depending on current sensory signals (Schilling & Cruse, 2020; Schilling et al., 2013). These controllers coordinate their behavior through, on the one hand, local coordination rules that influence the switching behavior (see Fig. 4). On the other hand, the different controllers are coupled through the body and the interactions with the environment. Actions of one leg affect other legs and can be sensed by those without requiring explicit information exchange (at least to a certain degree) (Dallmann, Hoinville, Dürr, & Schmitz, 2017). In this way, the system exploits the loop through the environment (Brooks, 1991). Decentralization of motor control is further supported by neurophysiological insights on the construction of the nervous systems, for example, in starfish (Heydari, Johnson, Ellers, McHenry, & Kanso, 2020).

3. Related work: Robot control approaches

Control of walking robots is a widely covered topic, and there are different general approaches. In this section, we will review related work with a focus on two common perspectives: First, a biologically-inspired perspective that aims for understanding how animals are able to produce robust adaptive behavior. Following the tradition of cybernetics, this is often considered a constructivist approach in which derived principles are tested in fully functional systems and are evaluated as well as compared with a focus on the resulting emergent behavior. Secondly, optimization-based approaches do not start or aim for a particular behavior, but instead explicitly aim for a well-defined objective (e.g. velocity). In particular, with the recent advances of machine learning and deep reinforcement learning approaches, such have become very popular.

3.1. Biological-inspired approaches to control of robot locomotion

A hierarchical organization of motor control (as introduced above) has been transferred onto many robot control architectures. In general, this is realized as a distinction between selection of actions and execution of actions on two—or more—different levels of a control hierarchy. But it is as well complemented by the development of more versatile legged robots that for example include elastic properties in the joint drives (Kim & Wensing, 2017; Schmitz, Schneider, Schilling, & Cruse, 2008; Semini et al., 2011).

Impressive work on applying a hierarchical structure to walking on rough terrain comes from the area of quadruped robots (Carlo, Wensing, Katz, Bledt, & Kim, 2018). There, the problem is divided and distributed onto different control levels of a hierarchy. While on a lower level the detailed movements of joints and motors have to be controlled, a higher level coordinates movements between different legs and mainly selects lower level control primitives. In many cases of locomotion control, on a higher level fixed gait patterns are assumed. Such feedforward control employing fixed gaits has been applied to multiple legged robots (Ijspeert, 2008). As one example for employing such a constant coordination in a hierarchical robotic control architecture, Kalakrishnan, Buchli, Pastor, Mistry, and Schaal (2010) used the LittleDog robot by Boston Dynamics to deal with challenging terrain. Spatial coordination is realized as a search for footholds: First, possible footholds are identified, and a rough path is planned using a pre-trained ranking function and a scanned three dimensional terrain map. During locomotion, the detailed path is executed depending on the current posture as well as the preplanned schedule. This approach produced quite stable walking over rough terrain. A similar approach has been applied recently by Bellicoso, Jenelten, Gehring, and Hutter (2018) on the robot ANYmal. But, importantly, these authors argue that different environmental situations also affect temporal coordination of legs on a higher control level. Therefore, they introduce a gait switching module that plans how to switch phases between fixed gaits. Such fixed gait patterns correspond well to fast walking and running in animals in which control is driven in a top-down fashion.

But, in normal walking situations control of locomotion in animals appears more sensory-driven. For example (as discussed above) experimental findings in insects show a wide diversity of walking behaviors that are better characterized as free emerging gaits and cannot be reduced to a set of fixed gait patterns (Bidaye et al., 2018; DeAngelis et al., 2019). Decentralization is assumed a key constituting principle of the underlying control system from which temporal coordination adaptively emerges in interaction with an unpredictable environment. It has been realized as a control principle in different legged robots. Such a notion of partially autonomous and concurrent control in distributed cluster can already be found in Brooks subsumption architecture (Brooks, 1986). Walknet is another recent example for such a control structure that follows a hierarchical and decentralized organization (Schilling & Cruse, 2020; Schilling, Hoinville et al., 2013). It has been realized recently on the hexapod robotHector (Dürr et al., 2019; Paskarbeit, Schilling, Schmitz, & Schneider, 2015; Schilling, Paskarbeit, Ritter, Schneider, & Cruse, 2021). Control is distributed hierarchically onto different levels. Each leg has its own decentralized controller that locally decides which action to perform depending on the sensed context. For locomotion, it is distinguished between two basic actions on the lower leg level (stance and swing movement). In forward walking, switching from a swing movement towards a stance action is initiated after the leg touches the ground and starts carrying weight. During stance mode, the leg contributes to carrying the body and propels the body forward until the position of the leg moves behind a posterior extreme position (PEP). Therefore, action selection is, on the one hand, sensory-driven and depends on the current local state of the controller. On the other hand, the six leg controllers coordinate their action, but again relying only through interchanging local information with the neighboring controllers. While Walknet is structurally quite a simple system, it is adaptive, showing free emergent gaits, and it can deal with severe disturbances (Dürr et al., 2019; Schilling & Cruse, 2017) as for instance loss of a leg (Schilling, Cruse, & Arena, 2007). Other approaches focus more on local intrinsic oscillations in distributed network structures of coupled oscillators. These have been applied successfully in robots (Steingrube, Timme, Wörgötter, & Manoonpong, 2010) often employing CPG-like structures (Ijspeert, 2008), e.g., in Inagaki, Yuasa, and Arai (2003) neural interactions of neighboring legs are driven depending on explicit gait patterns.

A central question in decentralized approaches deals with coordination of individual legs. Behavioral studies in stick insects provide a quite clear picture on the behavioral level to what extent neighboring legs influence each others' behaviors (Cruse, 1990). Forms of inter-limb coordination can be found throughout the animal kingdom (Dickinson et al., 2000) and have been applied, e.g., on a brittle star like robot (Kano, Kanauchi, Ono, Aonuma, & Ishiguro, 2019). Owaki and Ishiguro (Owaki & Ishiguro, 2017) realized an even further decentralized structure on a quadruped robot. In their case, coordination between individual leg controllers could be mediated without explicit coordination influences realized as neural connections. Instead, as legs are

connected through the body, they can directly experience and feel the contributions of the other legs for supporting the weight. This has shown to be sufficient for simple straight walking. While this further stresses the importance of taking embodied mechanisms into account, this particular form of coordination appears limited. More complex behaviors, as negotiating a curve, require in addition a form of spatial coordination of legs (Schilling & Cruse, 2007; Schilling, Paskarbeit, Schmitz, Schneider, & Cruse, 2012). In insects, a load-based mechanism appears mostly to support local control and reflexes (Akay, Ludwar, Göritz, Schmitz, & Büschges, 2007; Zill, Schmitz, & Büschges, 2004).

Forms of decentralization are further relevant for modular robotic approaches. In the case of legged, modular robots such systems are constituted of interchangeable leg modules which allows to reconfigure the anatomy and appearance of the robot, e.g., deciding on the number of legs or the arrangement of leg modules. As a consequence, the embodiment of the robot can be adjusted to a given task. Kim, Alspach, and Yamane (2017) proposed such a modular system that can consist of up to six leg modules, but applied precomputed gait patterns for the different configurations that were driven by a central control system (Whitman, Su, Coros, Ansari, & Choset, 2017). Ha, Kim, and Yamane (2018) used reinforcement learning in such a system to learn walking behavior for individual modules which is comparable to our approach, but is not focussing on exchange of information for coordination. A drawback of approaches employing fixed gait patterns is that there is an increasing number of potential leg configurations which requires extensive computation in advance. Furthermore, with an increasing number of legs, the control effort for a central unit grows dramatically and, overall, while the body structure becomes more easily adaptable, it is difficult to envision how to adapt control and body together. Therefore, Hayakawa, Kamimura, Kaji, and Matsuno (2020) recently proposed the KARAKASA system which is composed of single-legged modules that can form legged robot structures. Control is decentralized as each module is equipped with a controller. Joined modules form a distributed control system in which information between neighboring modules is explicitly exchanged (about the assumed geometric configuration). This allowed to achieve static walking on a flat plane with different leg configurations.

3.2. Optimization-oriented and learning-based approaches to locomotion in robotics

Learning-based approaches usually aim for optimizing a given objective and offer as an advantage that they do not require manual setup of control structures or modules. Therefore, such approaches have been applied for quite some time in the area of control of locomotion. One nice example is the work by Cully et al. (2015) that also employed a hierarchical representation as part of an evolutionary algorithm aiming for robust locomotion of a six-legged robot, but which aims more at switching of behaviors and presupposes precomputed fixed gaiting patterns. In the following, our focus will be on Deep Reinforcement Learning (DRL) which has been established as a promising approach in the last couple of years (Arulkumaran, Deisenroth, Brundage, & Bharath, 2017; Neftci & Averbeck, 2019). Initial success of DRL was found in the area of computer game playing (Mnih et al., 2015). DRL has now been as well used in continuous control tasks and on robots as well (Levine, Finn, Darrell, & Abbeel, 2016). But still, the field is widely dominated by approaches that deal with simulation environments as it appears that the nature of such robotic real world tasks is quite different from those of playing computer games (Hwangbo et al., 2019). Real world application have to deal with much more—and more unpredictable—noise compared to simulations. This poses a problem in itself for any control approach, but in the case of DRL it further questions the underlying assumption of a stationary Markov Decision Process (Kurach et al., 2019). Secondly, DRL aims for very specific and narrow solutions trying to exploit a defined reward structure as good as possible. This can lead to a form of overfitting (Lanctot et al., 2017; Zhang et al., 2018) and unwanted optimization in order to only gain a minimal higher reward at the cost of robust and broad application of a behavior as required for adaptive behavior (Schilling et al., 2019).

There is extensive work on DRL of locomotion that has been realized in simulation as this allows to gather many samples in quite a short time. This is largely due to the wide availability of standard environments for DRL as can be found in the OpenAI gym which includes one four legged robot called the ‘Ant’ (eight degrees of freedom). DRL has lead to well performing solutions in such well controlled environments, but generalization has shown to be problematic for such DRL-based approaches as learned control policies tend to overfit towards a given reward function or the specific task at hand (Zhang et al., 2018). One counter measure is to train agents in a broader variety of different environments and to increase difficulty of tasks over training time. In curriculum learning, the simulated robot is facing more diverse and changing environments over time (Heess et al., 2017). This has lead to more robust controllers that walk well in a range of seen environments, but it required intensive interaction time with the environment.

While such end-to-end learning approaches can produce robust locomotion controllers for environments that were seen during training, the results still show a tendency to overfit as pointed out recently by Tsounis, Alge, Lee, Farshidian, and Hutter (2020). Application on tasks that are out of the training distribution are still problematic, and the resulting approaches generalize badly towards novel environments as well as training appears quite inefficient. Among others, Merel et al. (2019) proposed several bio-inspired principles that could be employed in DRL and lead to more adaptive and faster learning control structures. In particular, they emphasize the hierarchical structure of motor control. Hierarchical DRL approaches help to deal with particular tasks that span longer timescales or in which reward is sparse (Kulkarni, Narasimhan, Saeedi, & Tenenbaum, 2016). In simulation, different approaches to locomotion already applied hierarchical structure (Frans, Ho, Chen, Abbeel, & Schulman, 2018; Heess et al., 2016; Peng, Berseth, Yin, & Van De Panne, 2017). In such cases, two levels of control were distinguished and operated on different, fixed time scales. This realizes a form of temporal abstraction. Such hierarchical approaches allowed to flexibly switch between distinct subtasks which enables dealing with a variety of distinct contexts (Frans et al., 2018; Peng et al., 2017). As one example, hierarchical DRL showed well suited for switching between obstacle avoidance, following a wall, or straight walking (Heess et al., 2016). Overall, hierarchical organization—as advocated by us as well (Schilling & Melnik, 2018)—showed in particular effective for learning building blocks that can be combined in longer sequences of actions. This is required in more complex tasks, as for example navigation, in which control should be learned from basic movement primitives. Transfer is, in such cases, realized as a reuse of basic motor primitives, and the hierarchical organization is understood in the sense of switching between different motor primitives. But it does not address dealing with variability through robust behavior as such and adaptation on a short time scale, as targeted by our approach, e.g., as in the case of an insect climbing through a twig which only provides sparse footholds and that can move unpredictably (Schilling, Hoinville et al., 2013).

Tsounis et al. (2020) proposed a particular hierarchical approach called DeepGait. DeepGait is constituted of model-based

planning on a higher level that is preplanning exact placement of sequences of footholds in a detailed acquired three-dimensional map of the surrounding area (we will not deal with other model-based approaches here that aim on planning or combining planning with control, see Lipson (2019) and Neftci and Averbeck (2019)). On a lower level, gaits are controlled as realizing sequences of footholds in order to keep the robot stable. In a sense, the architecture is following a similar idea as found in the area of more classical hierarchical control approaches. In contrast to the approach advocated by us, a higher level explicitly plans footholds and determines gaits and stepping patterns. From our point of view, this is in conflict with insights from biology which show—especially for slow walking—that gaits emerge in interaction with an unpredictable environment. In DeepGait, the different levels are trained using DRL as policies are represented as neural networks. This leads to a variety of locomotion capabilities, e.g. climbing stairs which nicely demonstrates how DRL-based approaches are able to leverage a hierarchical organization principle towards more difficult terrains. As one further difference, DeepGait relies on a detailed three dimensional map of the surroundings. In contrast, adaptivity in animals on a short time scale appears to be mostly driven by proprioceptive signals that allow for fast reaction times. Relying only on proprioceptive information, in Azayev and Zimmerman (2020) a hierarchical DRL approach is used. Specific lower level control modules were trained in different environmental settings, e.g., passing different types of terrain. On a higher level, these control primitives were switched by a recurrent policy level. Azayev and Zimmerman (2020) differentiate specific types of adaptation: On the one hand, micro-adaptation on the lower-level should guarantee robustness of behaviors. On the other hand, macro-adaptation deals with selection of appropriate behaviors or sequencing of these. This distinction coincides largely with our notion of adaptation across temporal scales. There is one key difference as, in their view, macro-adaptation is tasked with defining and learning gait patterns for different specific situations.

Concerning local and modular controllers, there has been some recent work. First, there is the very nice example by Huang, Mordatch, and Pathak (2020) in which one single policy is learned for the control of every individual joint. This policy is trained using a form of a weight sharing approach as a general policy over all joints which—in their case—is not only trained on one single robotic structure, but on quite different types of robotic creatures. Therefore, the learned policy encapsulates quite diverse joint movements including simple targeted movements as well as cyclic motion. Crucially, the input to the joint controller is local sensory information and information passed from other joints. While the content of the messages passed between joints is learned as well, the structure is given by the kinematic structure of the robot. As information is accumulated in two passes—one collecting information at a root node of the kinematic structure and, in a following step, sending this information back to all connected joints—this message can be tuned to contain global as well as local information. While this leads to impressive results of generalization of motion between different robot morphologies, learning took quite a long time which might be due to the central collection of all information in the root. In a similar direction, Wang, Liao, Ba, and Fidler (2018) exploit the structure of the body which they interpret as a graph. Using an explicitly defined graph structure, they explicitly learn which information should be shared across the graph. As a second group of methods, decomposition of the Reinforcement Learning problem has focussed on decomposition of the state (Laversanne-Finot, Péré, & Oudeyer, 2018) or the reward function. In reward decomposition, the total reward is composed of sub-rewards which are known to facilitate faster learning (Schneider, Wong, Moore, & Riedmiller, 1999). Early work focussed mainly on how to leverage such sub-rewards into finding better policies (Sprague & Ballard, 2003). Recent work focussed on finding a decomposition of rewards (Seijen et al., 2017) or more recently, for example (Lin et al., 2019), aiming at a decomposition that does not require prior knowledge. In a subsequent publication (Lin et al., 2020), they further aimed at disentangling the state space while uncovering a reward decomposition. But these methods have mainly been applied to the ATARI-type game-like or tabular scenarios. Third, there is work which does not rely on gradient-based learning, but employs local Hebbian learning rules. Najarro and Risi (2020) recently presented their work on evolving learning parameters in OpenAI's quadruped walking task. They were able to efficiently evolve a huge number of these learning parameters that parametrize local learning of the typical control policy and that allow for fast adaptation of the control policy. This allows for online adaptation on the agent. As an important difference, they evolve parameters for local Hebbian learning that show successful online adaptation afterwards, but the scope of the control policies remains global.

For application on real robots, most DRL approaches use simulation environments for pre-training controllers which are afterwards transferred to the robot (e.g., Hwangbo et al., 2019). This usually involves techniques for training on a wide range of environments (for example, using domain randomization (Mozifian, Higuera, Meger, & Dudek, 2019) or curriculum learning), or using variation of parameters describing the dynamics of the robot and its' interaction with the environment (e.g., dynamics randomization Tan et al., 2018 or, recently Peng et al., 2020 learned a low dimensional latent space of dynamic parameters during training in simulation that allowed fast transfer to the real robot which is called domain adaptation). There is only little work on directly learning a policy on a walking robot. In one recent example, Ha, Xu, Tan, Levine, and Tan (2020) introduced a safety constraint into DRL that reduced falling over and which did not require manual intervention for resetting of episodes, but instead relied on switching between different available tasks. Overall, application on robots is still quite challenging as DRL is usually quite data hungry, and acquiring data on real robots is difficult or potentially harmful for the robot (Chatzilygeroudis, Vassiliades, Stulp, Calinon, & Mouret, 2020).

Summary

There is interest in locomotion as an example for motor control in order to understand mechanisms of adaptive behavior in biology, and, from a machine learning perspective, in order to analyze how learning transfers to challenging, unpredictable real world tasks. Peng et al. (2020) contrast the different kind of approaches pointing out that, on the one hand, handcrafted controllers are difficult to scale. Learning can provide an already proven solution to extend skills towards much more agile tasks. On the other hand, Peng et al. (2020) argue that learning of control requires guidance, a form of inductive biases or priors (Chatzilygeroudis et al., 2020; Lake, Ullman, Tenenbaum, & Gershman, 2017) in order to lead to robust behavior. While hierarchical organization is already successful applied in learning-based approaches, we aim to leverage decentralization—as a well-established principle in control of locomotion—in a learning-based approach towards more agile locomotion and adaptive behavior. Learning of decentralized control structures promises to allow for much faster learning due to smaller search spaces over local information. Furthermore, in decentralized control behavior emerges out of interaction of different control (sub-)units which has shown advantageous when dealing with unpredictable tasks or for transfer between tasks. We are therefore interested in the question of how decentralization affects the learning process, which level of decentralization provides the best synergy of robustness and flexibility, and, last, in how far decentralized DRL enables a rapid adaptation to new terrain.

4. Methods

In this study, we wish to analyze how decentralization of the control architecture affects reinforcement learning for the important case of locomotion behavior. We consider the case of a four-legged simulated robot. We will compare different architectures that systematically vary the number of concurrent controllers, decompose the reward structure differently, and vary the scope of input information, analyzing how these factors affect the learning process, the achieved final performance, and the robustness of behavior in altered task settings. After a brief overview of reinforcement learning and how it is applied to different decentralized control architectures, we describe the specific four-legged robot environment and explain the details of the underlying simulation and learning frameworks that are used in all of our experiments.

4.1. Deep reinforcement learning control architectures

Reinforcement learning overview

Reinforcement Learning is characterized by an agent interacting with an environment and directly learning from such interactions. As the agent is producing actions, it is getting in response, first, an updated state S of the environment as input to the controller and, second, a reward signal that is used for learning. The goal for the agent is to learn over time a policy $\pi(S)$ for sequential decision making that maximizes the long-term return. Sequential decision making can be formalized as a Markov Decision Process defined as a tuple of:

- observable states $\mathcal{S}$ – which are proprioceptive sensory information as well as some information on body orientation and height and information on last actions (for details on scope of sensory information, see below);
- possible actions $\mathcal{A}$ – which are movements of the joints for our case;
- a reward signal $\mathcal{R}$ providing reward after choosing an action from the current state and integrating costs (e.g. for producing movements)—we will reward fast and efficient locomotion;
- transition probabilities $\mathcal{P}$ that describe the probability distribution over states after following an action from the current specific state, and
- a discount factor γ that describes how to decrease the weights of future rewards.

During learning the agent has to explore the state space in order to uncover rewarding states and rewarding actions depending on its current state. The agent therefore has to balance exploiting already known information in order to reach known rewarding states and explore unknown parts of the state space and consequences of decision that might provide additional or higher rewards (for an introduction to Reinforcement Learning see Arulkumaran et al., 2017; Sutton & Barto, 2018).

In the case of large or continuous state spaces, the overall state space cannot be searched exhaustively, in particular when dealing with high dimensional input and action spaces. Therefore, function approximation is used to learn approximative policies over the state space. In DRL, deep neural networks are used as a function approximator that maps a currently observed state to actions or action probabilities (Arulkumaran et al., 2017).

Decentralized control architectures

What is the impact of decentralization on learning of robust locomotion behavior? To study this question, we investigate a spectrum of architectures, ranging from completely central to fully decentralized architectures. Our strategy of investigation will be to compare different types of controllers that vary along three major feature characteristics (Fig. 5, and Table 1): First, with respect to concurrency of control—splitting the action space ($\mathcal{A}$) of the whole agent into separate subspaces with their dedicated controllers, e.g., one controller for each leg. Second, and closely linked with the first dimension of decentralization, is the structure of the reward signal ($\mathcal{R}$) used for training and how this might be decomposed. In particular with respect to costs: Movement costs are directly related to decentralized control units and can be locally differentiated as well, taking into account costs only for the joints controlled by that particular decentralized control unit. As a second characteristic, we analyze the effect of decomposition of the reward structure according to the decentralization and structure of the controller. Third, we vary and consider the scope of input information for each controller—splitting the state space ($\mathcal{S}$), e.g., using only local information for a decentralized controller from the controlled leg. Importantly, this is only sensible in the case of decentralized controller as we are considering information across a spectrum from, as a minimal input scope, only the controlled parts of the body up to including information from other parts of the body, e.g., a neighboring leg. For details on all eight different architectures, see Table 1.

On one end of this control architecture spectrum—serving as a baseline—is the standard approach as found in DRL which we will call a **centralized, global** architecture. For a centralized, global architecture, there is a single controller that has access to all the available information and for which training is driven by a global reward signal.

On the other end of the spectrum is a fully decentralized control architecture, with one independent controller for each

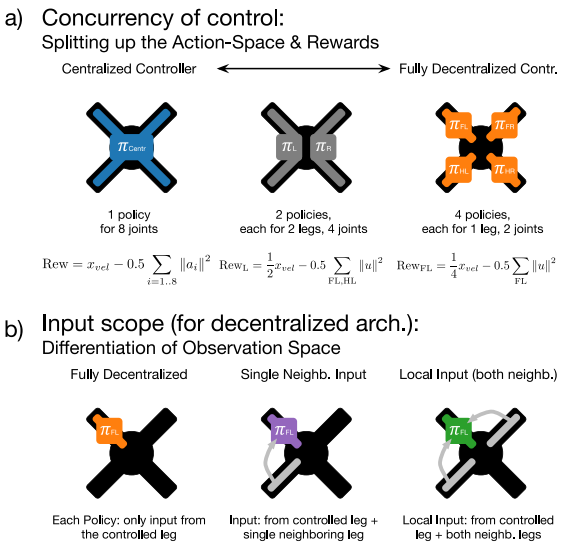

Fig. 5. Overview control architectures used for the four-legged simulated robot. As a first differentiation, we distinguish concurrency for which the spectrum of different architectures is shown in (a). In the centralized case (left), a single controller relying on all available information learns how to control all eight joints. Training is driven by one combined reward (shown is in all cases a simplified reward, not included are the external forces). Control can be split into multiple concurrent instances, e.g. one for each side (shown in the middle) or one for each leg (shown on the right, fully decentralized case). As a consequence, reward information can be more granular, e.g., only using costs associated with the joint movements of that particular leg. For a decentralized architecture, we can further distinguish the scope of information used as input to the controller (shown in b). In the fully decentralized case, the single leg controller only uses information from that particular leg and some global information (left). This scope can be broadened to include information from another leg as well (middle, there are further variations for which additional leg is providing information, not shown) or from both neighboring legs (right, called local information). Colors correspond to colors used in result sections for different types of architectures.

Table 1
Description of the different control architectures: Overview of differentiation of action space, observational state space, and the reward function for the different architectures (as used in the first experiments; for the last experiment, target velocity was added as an additional input state; for reward in the last experiment see Table 2 that compares different reward functions and parameters). The given decomposed reward function is related to the decentralization (only taking local costs of the controlled part into account). For the decentralized case, the overall reward is the sum of the individual rewards of all controllers (therefore, we have to scale the positive velocity dependent part to the number of controller, the local costs only appear once). In experiment 1.3, we compare using a decomposed reward for decentralized approaches compared to relying on the global reward for all decentralized controller (as given in the row of the centralized control unit). The different parts of the state space are detailed in the supplement, Table S2. Legs are abbreviated as: FL — front left leg, HL — hind left leg, HR — hind right leg, FR — front right leg. $\vec{f}_{ext}$ are the external impacting forces.

Controller Arch. Name	Number Contr.	Policy π (Action space)	State space obs_x	State dim.	Decomposed reward fct.
Centralized	1	$\vec{a} = \pi(obs_{\text{all}})$, 8-dim.	global/all inf.	43	$R = x_{vel} - 0.5 \left\|\vec{a}\right\|^2 - 0.05 \left\|\vec{f}_{ext}\right\|^2$
Each side (A: FL,HL, B: FR,HR)	2	$\vec{a}$ = conc ($\pi_A(obs_A)$, $\pi_B(obs_B)$) each $\pi(obs_x)$, 4-dim.	inf. from both controlled legs	27	$R_A = 0.5x_{vel} - 0.5 \left\|\pi_A(obs_A)\right\|^2 - 0.05 \left\|\vec{f}_{ext\_A}\right\|^2$
Diagonal legs (A: FL,HR, B: FR,HL)	2			27	
Decentr., local inf. incl. neighb.	4	$\vec{a}$ = conc ($\pi_{FR}(obs_{FR})$, $\pi_{FL}(obs_{FL})$, $\pi_{HL}(obs_{HL})$, $\pi_{HR}(obs_{HR})$) each $\pi_x(obs_x)$, 2-dim., representing a single ctrl.	ctrl. leg + both n.	35	$R_{FL} = 0.25x_{vel} - 0.5 \left\|\pi_{FL}(obs_{FL})\right\|^2 - 0.05 \left\|\vec{f}_{ext\_FL}\right\|^2$
Decentr., + single neighboring leg	4		ctrl. leg + neighb. (ccw.)	27	
Decentr., + single diagonal leg	4		ctrl. leg + diag.	27	
Decentr., single towards front	4		Towards front, at back	27	
Fully Decentralized	4		Only controlled leg	19	

leg. In this case, each controller has access only to local information, for example, from its particular leg and some specific global information (height of the body, orientation, and velocities of the body; for details on the state space for each controller, see supplement, Table S2). The **fully decentralized** architecture for the quadruped robot, therefore, consists of four concurrent controllers each only relying on information of the controlled leg. For this case, learning can be driven by a decomposed reward that consists of the overall positive reward from locomotion, but only the local costs of the controlled leg.

We further systematically distinguish in-between cases (Fig. 5(a)): First, along the dimension of concurrency, there are two control architectures that consist of two concurrent controllers, each controlling a different leg pair. In the first architecture, there is a controller for each **side**—one for the two left legs ("left" with respect to initial body orientation that is facing in the rewarded x-direction) and one for the two legs on the right side. In the other control architecture, each controller controls a **diagonal** leg pair (action and state spaces are detailed in Table S2). Again, for both these cases, reward can be decomposed into a global velocity component (in x-direction, as in all other cases), and **the local costs** of the controlled leg pair.

For the decentralized case, we distinguish overall five different arrangements (Fig. 5(b) shows three of these, all are depicted in Table 4, left column): All have in common that they consist of four concurrently running controllers—one for each leg—and that during training each controller is driven by a local reward signal (Fig. 5(a), bottom row shows reward signals). The architectures differ with respect to the scope of the state/input space. In the **fully decentralized** case, as mentioned, only information from the controlled leg is available. For the next three architectures, information from one further leg is integrated and made available to the decentralized controller. First, each controller has in addition access to information (joint angles, joint velocities, torques, previous control signals) from one **single neighboring** leg (considered counterclockwise when seen from above). Second, each leg has access to information from the **single diagonal leg**. The third variation is inspired from considerations from biology (Cruse, 1990): The front legs use as input information from the respective hind legs, and the hind legs interchange information (**single to-front**). Next, the last architecture integrates **local** information from both neighboring legs (for definition of control policies see Table 1, for specification of action and state spaces see Table S2 and Fig. 5).

Deep reinforcement learning algorithm: Proximal policy optimization

The different control architectures are trained using Proximal Policy Optimization. While early success of DRL relied on training an action-value-function using Q-Learning (Mnih et al., 2015), for continuous tasks often policy gradients methods are applied in which the policy as such is described as a function of the state. In policy gradient methods, an expectation of the gradient of expected return with respect to the policy parameters is derived from trajectory samples that were obtained while following the current policy in the given environment. The policy can be improved through gradient ascent by updating the policy towards higher returns. Importantly, the predictiveness of the estimate can only be guaranteed while staying close to the policy that was followed during sampling interactions. Therefore, one wants to enforce staying close to the current policy during the update step (Schulman, Levine, Moritz, Jordan, & Abbeel, 2015). One widely used policy gradient method is Proximal Policy Optimization (PPO) (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017) which ensures—during the update step—staying close to the previous policy by maximizing instead of the original a suitable "surrogate objective" that is derived from the policy function and the reward. PPO is an efficient estimation of policy gradients on collected batches of interactions and therefore widely used. As many policy gradient methods, it is implementing an Actor–Critic method which, on the one hand, directly represents the policy as a neural network that stochastically selects actions depending on the current state. On the other hand, a value function realizes a critic providing a value estimate for states and helps reduce gradient variance.

In this paper, PPO is used to train each control unit (for hyperparameters see Table S3). For the centralized approach this follows the standard approach of learning a single controller over the whole state space. In contrast, in the decentralized case the individual controller are learned independently and with respect to the scope of the input space. This means, for example, that for the fully decentralized control architecture of the four-legged walker, there are four independent controllers that are independently trained using four PPO instances and each relying only on the local information from the controlled leg. Nonetheless, experience from the simulation environment is jointly collected for all learning processes and the decentralized controller act as independent agents in the environment each controlling a part of the simulated robot.

Table 2
Components of the reward for different experiments and in comparison to more structured rewards as often used. $\vec{f}_{ext}$ are the external impacting forces, $\vec{u}$ the control torques applied in the joints.

	Ant-v3	Heess et al. (2017)	Experiment 1–3	Experiment 4
Forward reward	$x_{velocity}$	$x_{velocity}$	$x_{velocity}$	$\frac{vel_t+1}{vel_t\|x_{vel}-vel_t\|+vel_t} - \frac{1}{vel_t}$
Control cost	$-0.5\left\|\vec{u}\right\|^2$	$-0.01\left\|\vec{u}\right\|^2$	$-0.5\left\|\vec{u}\right\|^2$	$-0.25\left\|\vec{u}\right\|^2$
Healthy reward	1.0	$0.05n_z$	/	/
Contact cost	$-5*10^{-4}\left\|\vec{f}_{ext}\right\|^2$	/	$-5*10^{-2}\left\|\vec{f}_{ext}\right\|^2$	$-25*10^{-3}\left\|\vec{f}_{ext}\right\|^2$

4.2. Experimental simulation environment: Four-legged walker

To analyze a broad variety of possible coordination schemes of decentralized controllers, we focussed on a four-legged simulated walker that is derived from the popular OpenAI standard 'Ant' environment (simulated in Mujoco). The simulated robot has four legs, each consisting of two revolute joints, overall leading to eight controllable degrees of freedom. Joint ranges are limited to ranges as required for locomotion. In general, the learning characteristics are well known, and it is considered today an easy task as it deals with stable, static walking. This allows for many training repetitions as needed for systematic comparisons of differently structured controllers.

This task is widely used as a benchmark task. But, often the focus is only on the corresponding reward function as defined as part of the environment which aims for fast locomotion. When inspecting typical high performing controllers trained on flat terrain, these do not produce coordinated walking comparable to the one found in animals (Liang et al., 2018). Instead, often only two opposite legs are used for a more jumping like form of locomotion while the other two legs are not moved and oriented orthogonal to the direction of the locomotion in order to provide stability from falling to either side. Such gaits are not comparable with coordination as found in animals. This discrepancy reveals that the simple standard Ant-v3 model fails to capture important constraints under which biological walking has to occur. One such unrealistic feature is the extremely light weight of the simulated robot as compared to the weight of a properly scaled real robot. We therefore adapted the environment as we are interested in the resulting behavior as such and its biological properties.

Simulated robot: The weight of the simulated robot is set to 8.787 kg. This is one magnitude larger compared to the weight of the original 'Ant' environment, but brings the mass into a realistic range for a legged robot of that size. An increased weight has been applied as well in a similarly derived setting called 'roboschool' which lead to more coordinated gaits (Klimov & Schulman, 2017), in the sense that neighboring legs showed somewhat alternating phases. Increasing the weight has shown to make this a much more difficult learning task. Another approach that has shown to lead to more coordinated walking has been to use curriculum learning that groups a set of diverse tasks into "lessons" of increasing difficulty. We will use this strategy as well in later experiments.

State space: We restrict sensory inputs to ones that would be accessible on a real robot (for a detailed list see S1). The state space, therefore, includes information on joint positions, joint velocities, and torque measurements for each individual joint. As torque measurements, we access the 'passive force' for each joint which basically integrates external forces acting on the joint. This information represents a projection on the single dimension of the revolute joint which should be correlated with the external applied force. But while such egocentric information should encapsulate similar information, it is assumed to be harder to learn as it was found that Cartesian, three dimensional global representation can be easier to learn (Reda, Tao, & van de Panne, 2020). We further add two additional kinds of information. First, the current clearing (distance body to the ground), body orientation and velocities (in three dimensions plus three rotational ones) which is required for calculation of the reward function. Second, we in addition provide the applied control signals from the preceding time step as this is assumed central in theories of pattern oscillations, realized as forms of mutual inhibition (Ijspeert, 2008). In contrast, the ant environment usually is employing a much larger state space of 111 dimensions (which includes height, orientation, and velocities of body, joint angles and velocities, external forces acting on all parts) and uses external force calculations as provided by the simulation engine.

Reward structure: In locomotion tasks, one component of the reward is given by displacement in a defined direction. Typically, it is simply aiming for higher velocities. Usually, further cost terms are introduced. Our reward function consists of three components (see Table 2): First, a velocity dependent reward which rewards movement in one direction in global coordinates (x-direction; legs are named with respect to this defined direction, i.e., front legs are initially pointing into that direction). In the initial experiments, this positive reward is simply given as the velocity in x-direction. For the last experiment, the reward structure was adapted and locomotion aimed for a specific target velocity. Simply training for maximum velocity has lead to not-so-well coordinated walking behavior that, for example, includes more jump like motions and is often not stable (Azayev & Zimmerman, 2020). Therefore, we as well rewarded reaching a specific target velocity that corresponds to walking at different velocities (target velocity was used as an additional input to the controller). Reaching the target velocity was rewarded with a maximum reward of 1.0 per control step and linearly decreased for slower velocities (a not moving robot would gain zero reward). As the overall possible reward is much smaller compared to the other experiments, we had to adjust the coefficients of the control and contact penalty accordingly. They were set proportional to the change of the overall return and afterwards account for a similar portion of the return.

As a second component of the reward, a cost term penalizes effort of the actuators as a control cost in order to favor more energy efficient walking behavior. Third, another additional cost term penalizes contact forces as well.

In general, the reward structure follows the one given in the original 'Ant' environment. But, as one change, the 'Ant' environment contains an additional reward which is given for keeping the current simulation run alive (healthy reward). This has shown to be important in other environments, e.g., in a bipedal walker task, in which stability is much more of a challenge often ending simulation rollouts early on Heess et al. (2017). It is conditioned on the main body being kept in a certain height range (between 0.2 and 1.0 units). But has shown to be inconsequential for the four-legged robot case and was therefore left out. As a second change, we used a higher influence of the contact costs as otherwise—when using the original parameter—this would be negligible.

4.3. Implementation details

Code used for training the four-legged robot and running the simulation is provided together with exemplary trained controllers, videos, and results in the open repository https://github.com/malteschilling/ddrl.

Simulation environment

Experiments were run in simulation using the Mujoco physics engine (Todorov, Erez, & Tassa, 2012). We used Mujoco in version 1.50.1.68.[1] The simulation engine was run at a step size of 10 ms and the controller at a frequency of 20 Hz (this equals a frame skip of 5 which means that the controller send only every fifth dynamic simulation step new control signals, in-between the old signal is still active).

Ray and RLlib

As a framework for DRL we used Ray (Moritz et al., 2018) and RLlib (Liang et al., 2018). Ray is a framework that allows to efficiently distribute python tasks on clusters implementing an actor model for concurrent processing. It offers an easy and unified interface for running parallel tasks which are internally modeled as a graph of dependent tasks that can be executed. RLlib as part of Ray provides a scalable and efficient DRL framework which as well is centered around these actor-based computations that simplify setting up parallel computation and encapsulation of information. For our case this is crucial, as we consider concurrent and parallel control. This can be easily realized in RLlib as multiple (independent) actors—each representing one controller or sub-controller—that can be trained in a single environment and efficiently in parallel on a compute cluster. We use the provided PPO implementation as a DRL algorithm which follows code-level recommendations for PPO (Engstrom et al., 2020). For hyperparameters see Table S3 in the supporting informations which largely follows the recommendations from Andrychowicz et al. (2020) and Reda et al. (2020)

Deep neural networks

As explained above, neural networks are used in DRL for function approximation. In PPO, there are two different networks as part of the Actor–Critic approach (Fig. 6). Both networks receive the same inputs. On the one hand, there is a value function network that is used as a critic that provides a single value for the current state. On the other hand, the policy network directly parametrizes stochastic action selection through providing as an output, for each dimension of the action space, the two parameters of a normal distribution (mean and standard deviation). These two outputs are used to sample the next action for all dimensions:

Approximate π through a neural network f_θ :

$f_\theta(obs) = \mu_i, \sigma_i \forall$ controlled joints i

$\vec{a} \sim \pi(obs) \rightarrow a_i \sim \mathcal{N}(\mu_i, \sigma_i)$

For both networks—policy and value function network—we use the same structure for every controller. We use a simple structure of two hidden layers of each 64 units with *tanh* activation functions. As in the decentralized case multiple controllers are used, this might offer an advantage: When there are multiple networks of the same structure, these contain overall more units and weights which increases the possible capacity of the networks used in the decentralized cases. Therefore, we investigated in one experiment (see Section 5.3) the influence of the network size. We found, first, that the chosen size is appropriate for all different architectures, and, secondly, that the decentralized approaches—in contrast to the centralized approach—are robust to the specific setup of the neural network hyperparameters.

Inside RLlib, we used Tensorflow 2 for representation and training of the neural networks. The networks were initialized using Xavier (Glorot uniform) initialization (Glorot & Bengio, 2010) which we added to RLlib. Initialization aimed for action values with zero mean and a low standard deviation which has shown to improve learning speed (Andrychowicz et al., 2020; Rao, Aljalbout, Sauer, & Haddadin, 2020) in particular when using very small values for the standard deviation in the output layer.[2] Input dimensions differed for the different types of architectures and are described in Table S1.

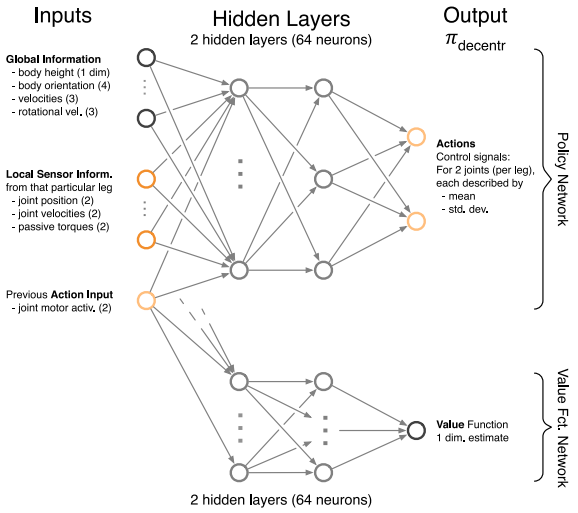

Fig. 6. A single controller (for control of a single leg) in the fully decentralized architecture, realized using PPO: It consists of two networks, one policy network (upper part) and a value function network (lower part). Both networks are realized as neural networks consisting of two hidden layers (each 64 neurons). Network setup differs for the different architectures (see Table S2). Shown is one controller for the fully decentralized case in which input consists of global information (body height, orientation, velocities) and local information only from that particular leg (joint positions, velocities, and passive torques as well as last control signal), overall this results in 19 input dimensions. While the value function network always only produces a single value estimate, outputs differ for the different architectures in the policy network. For the decentralized case, there are multiple such controllers that only control corresponding joints, e.g. four decentralized controller that each control a single leg consisting only of two joints. In contrast, a centralized approach with a global input scope would use information from all legs and one controller would be used to control all legs and all eight joints (for differentiation of control architectures see Section 4.1).

5. Results

Our main goal is to understand how performance, learning, and generalization depend on the specific control architecture,

[1] The current Mujoco version 2.0 introduced some changes that cause issues concerning the calculation of measured external forces in the robot limbs (see https://github.com/openai/gym/issues/1541). While we provide a fix enforcing the correct calculation of the external forces in version 2.0, we did not use these sensor readings in the end and introduced further changes that make this a more realistic and biological setting.

[2] In contrast to Andrychowicz et al. (2020), we found it quite advantageous to use Xavier initialization instead of a simpler random scheme—as used in RLlib—that only takes into account the shape of the outgoing layer (they follow Rao et al., 2020 who propose to simply use orthogonal initialization). One reason for this different finding is that in Andrychowicz et al. (2020) dimensionality of input spaces were not varied. As a result, the simpler initialization and Xavier initialization operate basically in the same way. In contrast, in our experiments input space dimensionality varies considerably and Xavier initialization is taking this into account which appeared as crucial in our test experiments. Xavier initialization was leading faster to stable learning results independently of the input space dimensionality.

Table 3
Overview of experiments and the research questions (experiment number equals subsection number inside this result section).

Exp.	Research question
1.1	Can decentralized controller perform on the same high level compared to standard centralized approaches?
1.2	How is learning speed affected by decentralization and locality of information?
1.3	How is learning speed affected by factorization of the reward signal?
2.1	How does the different controller generalize to novel, more difficult and uneven terrain?
2.2	How do the different controller compare with respect to efficiency?
3	How robust is training with respect to neural network hyperparameter variation?
4.1	How do the different controller learn and perform when aiming for coordinated walking?
4.2	How does the different controller generalize to variation of terrain?
4.3	How does the different controller generalize to variation of target velocity?
5	Which input information is used by a well-performing standard centralized controller?

the available input dimensions, and the structure of the reward. Therefore, we present multiple experiments and evaluations on learning of locomotion using the different architectures and testing these in different and increasingly more difficult contexts. In the first experiment, we analyze how decentralization of the control architecture, input scope of controller, and decomposition of reward affect the reinforcement learning process as such on flat terrain. Second, we evaluate performance and efficiency of learned controller when generalizing to novel terrain. Third, we compare variation of the neural network capacity and analyze robustness for the different architectures. Fourth, we change the reward and aim for a given target velocity while using a curriculum of increasingly uneven terrain during training. Fifth, as one further area of interest, in a post-hoc analysis we visualize the importance of input channels in a trained controller for the case that all possible inputs are available during training. The Table 3 provides an overview of our research questions and the different experiments and evaluations.

5.1. Experiment 1 – DRL for variation of the controller architecture for the four-legged robot

Overall goal of the first experiment: The first learning experiment aims to answer the question how does different control architectures—w.r.t. decentralization of the action space, the input scope of the observation space, and the decomposition of the reward, see Table 1—affect performance and learning over time.
Experimental procedure: We studied learning to walk on flat terrain for the simulated four-legged walker. The different architectures (see description in Table 1) were each trained 10 times using random seeds for 20 million environment simulation steps. Variation was done with respect to decentralization and input scope. Last, we consider the effect of reward decomposition in an additional experiment.

5.1.1. Research Question 1.1: Can decentralized controller perform on the same high level compared to standard centralized approaches after training?

Result — Performance of learned controller: All architectures converged over time towards a return of around 3000 (see Supporting information, Table S4) and all learning approaches were able to learn walking behavior (see supplemental video S1) at a high velocity. While all approaches were able to learn walking behavior, learning progressed quite differently for the different architectures (see learning curves in Fig. 7(a)). The centralized, global approach shows worse learning characteristics. First, return is increasing much slower compared to all the other architectures. Second, variance becomes very large for a considerable time of training. Considering slower learning, we allowed the centralized approach to learn for a longer time, but it converged to a similar return value when trained for overall 40 million time steps (see supporting information, Fig. S1).

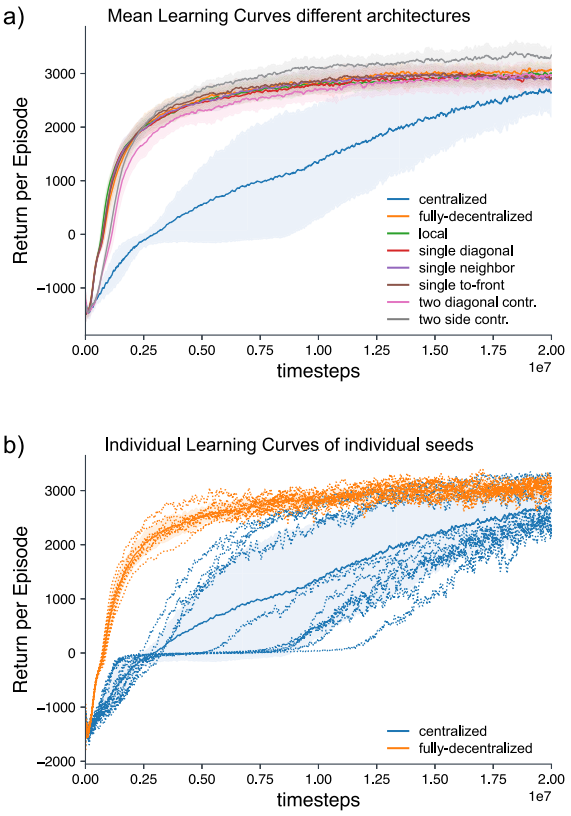

Fig. 7. Learning curves for different controller architectures over learning time (experiment 1): (a) Mean return per episode—calculated over 10 seeds—for all different learning architectures is shown during learning, given in simulation steps (interactions with the environment). Standard deviation between seeds is given as shaded area. (b) Individual learning curves for centralized (blue) and fully decentralized (orange) controller architecture over learning time, given in simulation steps (interactions with the environment) and shown is the return per episode. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.1.2. Research Question 1.2: How is learning speed affected by decentralization and locality of information?

From our first observation that learning progresses differently for the different architectures, we want to better understand which factors influences this learning behavior. Therefore, in a next step, we take a more detailed look on the learning phase from experiment 1.1.
Result — Comparing learning performance: The centralized approach shows a remarkably different learning characteristic (Fig. 7). In particular, there appears much more variance during learning for the individual trained centralized, global controller. Therefore, we visualized these 10 trained seeds individually (Fig. 7(b)). For comparison, only the fully decentralized approach is shown (in orange), the other architectures behave more similarly to this approach. Learning of all 10 fully decentralized controllers follows a common trajectory and there

is—for Deep Reinforcement Learning surprisingly—little variation. In contrast, for the centralized approach there are basically two different extremes of how learning progresses: On the one hand, learning continuously improves which is comparable to the other learning architectures. But, importantly, even in these best cases, learning still progresses much slower compared to learning in the other architectures (e.g. a mean return of 1000 over all 10 seeds is reached in the fully decentralized case already after 1.2 million simulation steps, while the best centralized approach requires 3.5 million steps). On the other hand, for many cases the reward is stuck for a long time around zero. This can be explained by the reward structure as we see different phases during learning: Initially, there is a strong negative return that is due to the costs associated with the actions. The controller learns to avoid unnecessary and costly movements (captured in the reward structure as the control cost, the summed squared action signals). During this phase, not moving at all appears as a good solution as it does not incur any costs. In a next phase, learning has to find movements that produce a large positive reward which at the same time overcomes the resulting costs of these actions. In a centralized approach, this problem is quite hard as all costs are lumped together, and it is difficult to distinguish which of the costs are necessary for the positive reward from locomotion and which point out unnecessary movements. This is a difficult credit-assignment-problem in the case of the centralized controller with global information and a single global reward. As a consequence, some centralized approaches struggle for a long time until (by chance) they find actions that produce a positive reward. Afterwards, these start to improve steadily as well. In contrast, for the decentralized control approaches, we used a decomposed reward in experiment 1.1 that can be split as well. The reward consists of local cost terms which means these only affect the reward of the controlled leg (see Fig. 5(a), bottom row shows the rewards).

To further access the learning performance, Andrychowicz et al. (2020) proposed a measurement that integrates returns during the learning over time. The learning performance is evaluated as the average score over time during the learning phase. It is evaluated as the mean of the episode return over learning epochs during that particular run—which is proportional to the area under the learning curve (Fig. 7). Fast learning approaches will show higher scores compared to approaches that take more time to reach a certain level (Andrychowicz et al., 2020). We apply this measure to further highlight the difference in learning over time (see Fig. 8(a)). In particular, we are interested in comparing the learning performance at the end of training: Fig. 8(b) shows the learning performance and variation at the end of training for all the architectures (detailed values see in Table S5).

We analyzed the learning performance of the unpaired samples from the different architectures using the non-parametric Kruskal–Wallis test (Kruskal & Wallis, 1952) (as the data appears not normally distributed) and for post-hoc analysis using the Dunn (Dunn & Dunn, 1961) post-hoc test (applying Bonferroni correction) following Raff (2019). The Kruskal–Wallis test showed that control architecture had a significant strong effect on how well the controller learned (H statistic $=$ 42.421, 8 groups, each 10 values, $p < .001$, $\epsilon^2 = 0.423$, $\eta^2 = 0.464$ which indicates the portion of variance in the dependent variable that is explained by the independent variable). The post-hoc test using Dunn's test with Bonferroni correction showed significant differences between the centralized architecture compared to other architectures. The centralized architecture showed a highly significant ($p < 0.01$) worse learning performance compared to the fully decentralized, local (sensory input from two neighboring legs), single neighbor, biological inspired (information from single neighbor, directed to the front), and the sidewise pair

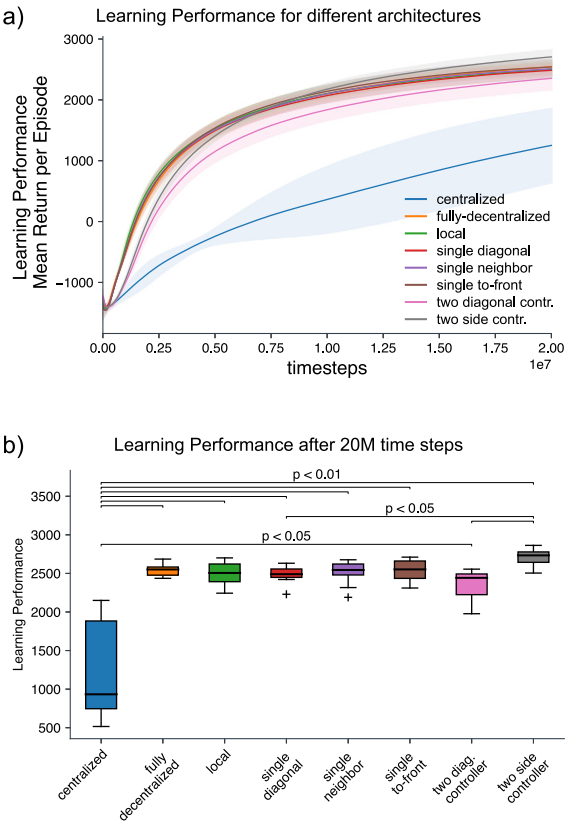

Fig. 8. Learning performance over learning time, evaluated as the average score over time during the learning phase. (a) shows learning performance over training time (y-axis). (b) Learning performance accumulated during learning at the end of training for different architectures. For each architecture, 10 individual training runs were performed. Shown is the mean learning performance over 20 million simulation steps. Learning performance, as proposed by Andrychowicz et al. (2020), differs from mean return during learning (as shown as learning curves in Fig. 7). The learning curves show the mean return at a specific point in time. Learning performance was especially introduced as a measure for learning up to a specific point in time. It is meant as a summary of all learning up to that point in time and is integrating the mean return over time which leads to a smoother curve. Learning performance is evaluated as the mean of the episode return over learning epochs during that particular run which is proportional to the area under the learning curves (Fig. 7).

of controller architecture. Furthermore, there was a significant difference between the centralized architecture and the single neighbor architecture which only used information from the diagonal leg ($p < 0.05$). Last, the pairs of controllers for diagonal legs showed a highly significant worse learning performance compared to the sidewise pair of controller architecture (for details see supporting information, Table S5, and Fig. 8).

Overall, these results confirmed our initial observations: Learning of a centralized control architecture performs significantly worse compared to the decentralized (four leg) architectures that use local cost structures and only rely on local information. The type of architecture showed a strong effect on learning performance.

5.1.3. Research Question 1.3: How is learning speed affected by the factorization of the reward signal and using local costs as part of the reward?

Considering the difference in learning, we argued that the decomposition of the reward structure is guiding exploration in the decentralized cases, leading to faster learning without getting stuck in local minima.

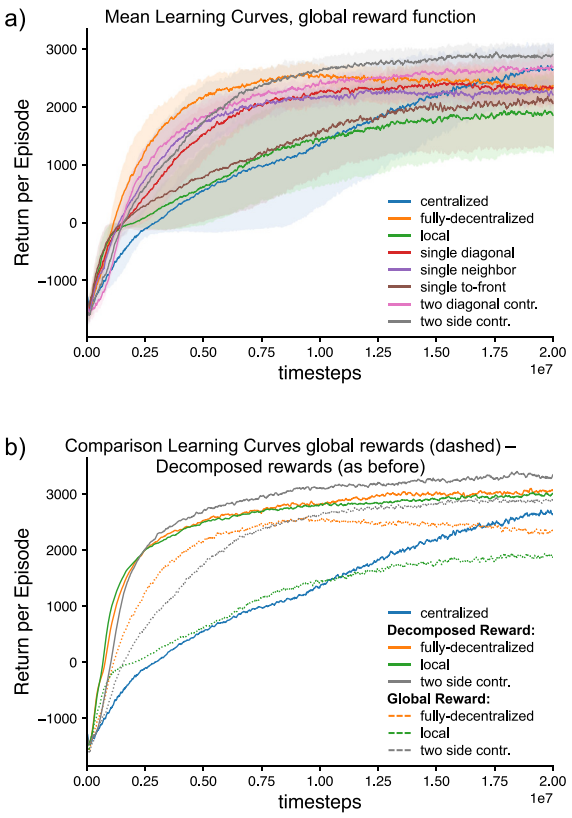

Fig. 9. Learning curves for different controller architectures over learning time (experiment 1.3) when relying on a global reward: (a) Mean return per episode—calculated over 10 seeds—for all different learning architectures is shown during learning, given in simulation steps (interactions with the environment). Standard deviation between seeds is given as shaded area. (b) Comparison of mean return per episode during learning for decentralized architectures using either the decomposed reward (as in experiment 1.3, shown as a solid line) or when using a single global reward for all controllers (dotted line).

Experimental procedure for global reward experiment: We repeated our training under the same conditions as above (training on flat terrain, different architectures w.r.t. decentralization and locality of input information), but changed the reward signal and used for all different architectures the same global reward signal. In case of the decentralized cases, this means that the positive reward and all costs were, first, globally aggregated and summed up. In a second step, this reward was simply distributed to the individual decentralized controllers without any local cost or reward information (reward was simply divided by the number of controllers in order to keep the overall reward comparable; this scaling had no effect on the PPO training Engstrom et al., 2020 which we ruled out through an additional experiment, described in supplement S-III).

Result — Influence of local cost structure: The results for 10 seeds for each architecture with a global reward are shown in Fig. 9(a). As a first observation, when using the same global reward for all different architectures, we still can observe that decentralization leads to faster learning. But the results vary considerably for the differing architectures. First, we observe more variance (shown as shaded area) in all cases. Still, the fully-decentralized case learns very fast and shows limited variation. When increasing the dimensionality of the state space (adding information of an additional leg, as is the case for the single diagonal and single neighbor architecture), learning is a little slower. The same holds true for the two architectures that consist of a pair of controllers which also depend on information of two legs and which learns at a comparable speed. All these approaches show only mild variation (shown as standard deviation in shaded area). But, this changes when dimensionality of the state space is further enlarged towards information from both neighboring legs. Variation becomes much larger, as in individual cases learning gets stuck in local minima (simulated robot not moving) for some time. As a result, the learning curve of the local controller is much closer to the centralized approach that uses all information as input and, in the end, is even performing below the centralized control architecture in the mean. It appears that the lower dimensionality of the state space is helping faster learning as it reduces the search space for exploration. The only exemption is the single to-front controller that is based on the state space of two legs, but performs worse than all the other approaches of the same dimensionality and is in the range of the centralized case.

Fig. 9(b) compares results for four selected architectures: On the one hand, learning curves are shown for using decomposed reward (as in Section 5.1) as solid lines. On the other hand, learning curves are provided as dotted line when relying on the same global reward signal. The decomposed reward helped in all cases to learn considerably faster. Furthermore, in the decentralized cases the decomposed reward function resulted in higher returns at the end of learning and better performing controllers when comparing these two conditions.

Summary — Influence of decomposed reward: The comparison of training controller with a decomposed reward in contrast to a global reward supports our argument that local costs considerably help learning.

Summary experiment 1:

The first learning experiment demonstrated that only relying on local information did not negatively impact the performance of trained decentralized controller on flat terrain (the terrain the controller was trained on). Secondly, we found that decentralized learning was significantly faster compared to the standard centralized case (which required three to eight times the number of simulation steps for reaching a similar level). Third, local cost structures for the decentralized control architectures showed to help guide the learning process and, in particular, the exploration process of DRL.

5.2. Experiment 2: Evaluation of trained controller — Generalization and efficiency

5.2.1. Research Question 2.1: How does the different controller generalize to novel, more difficult and uneven terrain? I.e. what is the influence of decentralization and locality of information on behavior generalization performance?

Goal: In a next step, we evaluated how the different trained control architectures perform when tested on uneven terrain (Fig. 1(a)). Uneven terrain poses a novel problem for the trained controller. Therefore, we used this to test for generalization and robustness.

Approach: Uneven terrain was generated using the bumpy terrain generator from the DeepMind control suite (Tassa et al., 2018) which is realized inside the simulation as a height field. The terrain was randomly generated and consisted of overlaid sinusoidal shapes which are parametrized by the spatial scale (we used the pre-defined value) and a smoothness factor (ranging from 1.0 for a smooth, flat plane towards 0.0 which would generate a maximal bumpy terrain). We systematically varied the smoothness of the surface from a smooth surface (1.0 which is the same as the setting during training for flat terrain) towards more bumpy surfaces (smoothness factor of 0.9, 0.8, 0.7, 0.6) until the controller were struggling to produce locomotion any further. For each controller architecture, each of the 10 different

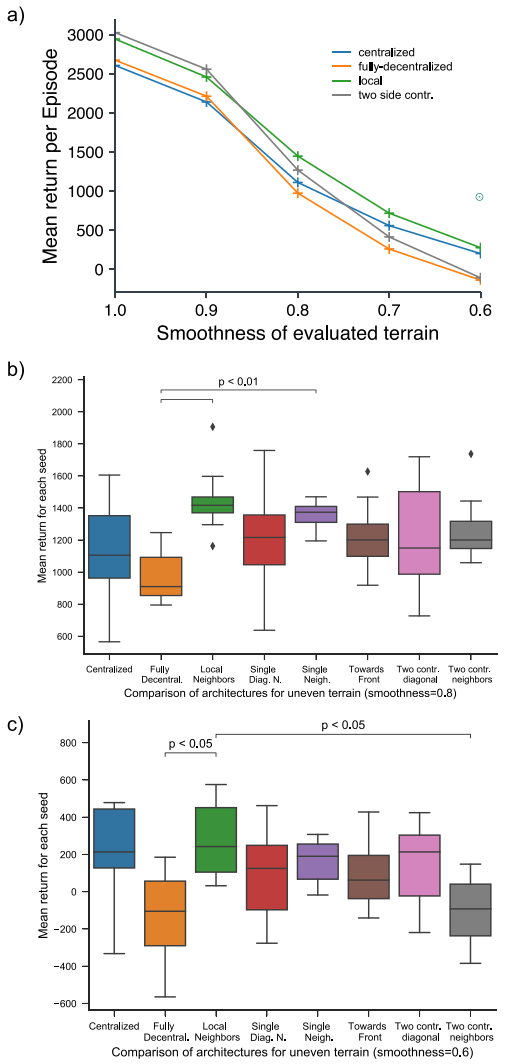

Fig. 10. Evaluation on different types of terrain: For each architecture, all trained controller are evaluated for 100 runs on more and more uneven terrain (smoothness 1.0 is flat terrain and 0. would be very bumpy terrain). In (a) mean return for four selected architectures is shown and how performance decreases for more difficult terrain. For comparison, for the most difficult terrain the best found expert policy (i.e. trained on the bumpy terrain) is shown as a dotted, green circle. (b) and (c) compare mean returns for two specific selected types of terrain (smoothness = 0.6 and 0.8 respectively) for the different architectures. The fully decentralized architecture performs significantly worse compared to the local, decentralized architecture.

trained seeds was tested for 100 simulation runs of 1000 control steps (equaling 50 s). Mean results for the different controller architectures are shown in Fig. 10.

Result – Generalization capabilities: As a first observation, we see that mean return per episode is going considerably down when facing more difficult terrain as is to be expected. Importantly, this is not alone due to the difficulty of the task, as expert policies that were trained on uneven terrain showed better performance (see Supplement, Fig. S3; best performing architecture when trained on uneven terrain was the local architecture with a mean return of 924.39 when evaluated on a height field of smoothness 0.6; worst performance was found for the fully decentralized architecture with a mean return of 714.92).

Secondly, when considering the different control architectures, we further observe that they show different generalization capabilities. This, in particular, differentiates the different types of decentralized architectures. It appears that an important factor is the scope of information used by the control architecture: The global, centralized architecture is gaining compared to some of the other approaches, in particular compared to the fully decentralized approach. Furthermore, in particular the local approach—a decentralized approach that nonetheless is integrating information from two neighboring legs—shows the best performance on more difficult terrain. It performs highly significant better compared to the fully decentralized approach for an intermediate uneven terrain, and significantly better for even more uneven (bumpy) terrain (Fig. 10(b) and (c)) in which the fully decentralized approach is not able to produce any positive mean return anymore. It appears that for more challenging terrain more sensory information becomes advantageous and important for behavior. This is further supported as the decentralized approach that integrates information from one neighboring leg significantly outperforms the fully decentralized approach for intermediate difficult (uneven) terrain. Further, note how narrow these two distributions (of the decentralized local and single neighbor architecture) are. While some amount of information appears helpful, for the centralized approach we observe that the distribution is much wider (10(b) and (c)). It appears that in the case of the centralized approach some of the architectures are very bad at generalization.

Summary: The scope of input information (the observation space) of decentralized controller showed to be important for generalization performance which we evaluated as robust behavior on uneven terrain. Information from neighboring leg(s) appears as crucial information for coordination in uneven terrain. But already local information from only two neighbors was sufficient to produce well performing controller. It appears therefore as unnecessary to include all available information from all legs as these appear as redundant.

5.2.2. Research Question 2.2: How do the different controller compare with respect to efficiency and velocity?

Goal: We want to compare the different trained controller with respect to individual behavioral measurements instead of using the return signal that accumulates different contributions. How good is the learned behavior, measured as velocity? And how energy efficient are the different walking behaviors?

Approach: DRL is optimizing towards the given reward function which in our case integrates different components of positive reward (as velocity in x-direction) and control as well as contact costs. We used the evaluation runs (Section 5.2) to disentangle the different contributions and compare the performance with respect to different characteristics (see Table 4). As described in the previous section, results were collected on 100 evaluation runs of each 1000 control steps for each of the ten different seeds for each of the eight different control architectures. When test runs terminated early, due to the robot flipping over, we accounted for that in the calculation of velocity. Evaluation was run on different types of terrain. First, we are interested in forward locomotion and measure this as the resulting velocity. Secondly, we want to measure the efficiency of the resulting gaits. This can be assessed using the Cost of Transport (CoT) which is defined as the ratio of the average power consumption with respect to the product of velocity and weight (Gabrielli, 1950; Ijspeert, 2014):

$$CoT = \frac{P}{mgv}$$

The power consumption in simulation is calculated for each joint as the torque acting along the joint movement, i.e. the product of torque multiplied by the velocity of that joint. Calculation of individual cost of transports for each run and afterwards averaging would have the drawback that runs with a low velocity

Table 4
Evaluation of different control architectures. Each architecture was trained 10 times and each of these seeds was evaluated for 100 episodes on multiple different terrains (flat terrain; uneven terrain, smoothness 0.8; bumpy terrain, smoothness 0.6). Given is the mean return, mean velocity, and cost of transport over all episodes per architecture for a specific terrain type. Shown are the different control architectures (figures highlight decentralization and information available in each controller — white lines signify one controlled entity, e.g. in the decentralized case a single white leg means that each leg has its own controller; gray illustrates the information that are available to that single shown controller, e.g. for local sensory information from a neighboring leg, a leg controller has access to information from itself and the in gray shown legs; arrows indicate how information is shared between other legs).

Configuration		Flat terrain			Uneven terrain (0.8)			Bumpy terrain (0.6)		
		Return	Vel.	CoT	Return	Vel.	CoT	Return	Vel.	CoT
Fully Decentr. (4 controller)		2673.0	3.33	6.283	974.0	2.15	7.424	-137.5	0.64	14.099
Decentr., inf. single neighbor		2874.8	3.48	6.244	1358.7	2.37	7.967	165.9	1.07	14.379
Decentr., inf. diagonal leg		2868.4	3.47	6.137	1185.1	2.37	7.420	86.7	0.90	12.341
Decentr., inf. towards front		2820.7	3.43	6.056	1229.5	2.32	7.594	97.2	0.93	13.033
Decentr. local inf., both n.		2943.4	3.54	6.169	1447.0	2.55	7.283	276.4	1.20	11.357
Split (2) contr. each side		3025.6	3.85	6.758	1268.3	2.60	8.391	-108.6	0.88	14.959
Split (2) contr. inf. diag. legs		2786.6	3.51	7.021	1200.9	2.43	8.467	143.3	1.14	13.112
Centralized single contr.		2605.9	3.57	8.224	1110.5	2.55	9.324	201.5	1.36	12.641

close to zero would have very high CoT that would dominate the average. Such low velocities occur with increasing difficulty of terrain more often, as the robot got stuck from time to time and was not able to move any further forward. These runs would be highly overvalued in the efficiency calculation and the CoT in these cases was mostly dependent how early the simulated robot encountered a bump in the terrain, it could not climb. Therefore, we calculated mean cost of transport as the mean of power consumption over all runs divided by the mean velocity of the robot over all runs for that particular architecture.

Result – Efficiency of learned locomotion: When looking at the results for the different architectures, we see that the architectures all reached a similar mean velocity on flat terrain. The fastest velocity (by a small margin) was reached by the split controller which consists of two controllers, one for each side (left, right). The centralized and local architecture (four controllers, integrating information from both neighboring legs) follow with respect to velocity. The fully decentralized approach was the slowest. With respect to energy efficiency (Arena, Patanè, & Taffara, 2021), the decentralized architectures actually performed much better and were considerably more energy efficient—measured as cost of transport—when compared to the centralized approach. A detailed analysis of the resulting behavior and differences between the controller from a behavioral perspective is beyond the scope of this article. We analyzed emergent gaits and produced footfall patterns for the different architectures (see supplement S-VII), but these provide more a qualitative view onto the emerging behavior and did not show considerable differences.

When turning towards generalization to uneven terrain, we, first, observe that velocities went down and cost of transport increased. This was true in particular for the fully decentralized approach which was walking at a much slower velocity compared to the centralized or local architecture (around half the velocity). In general, energy efficiency should be expected to get worse for uneven terrain as the robot has to climb up and down through the small hills in the environment. All approaches appear to struggle with this task. With respect to the required energy, we observed that the local controller performed relatively well: It still produced a reasonable velocity and the cost of transport was the lowest (for the intermediate uneven terrain, the local controller's cost of transport is even smaller compared to the centralized approach on flat terrain). Other decentralized approaches start to produce considerably higher costs of transport.

Summary: Decentralized controller and centralized controller were all able to produce fast walking on flat terrain (on which they were trained) which is in agreement with the results of experiment one. But there was a considerable difference in efficiency. Decentralized control architectures appear to be more energy efficient (producing lower cost of transport). When turning towards uneven terrain (not used during training) including information from neighboring legs as input to the controller showed important, but again it was sufficient to use information from two neighboring legs. As a result, a decentralized local controller that integrates information from two neighboring legs showed to generalize relatively well to uneven terrain and was the most energy efficient on the more difficult terrains.

5.3. Experiment 3 – Variation of neural network model hyperparameters

Research Question 3: How robust is training with respect to hyperparameter variation? In particular, considering the selection of hyperparameters of the neural networks and the possibility of overfitting?

Goal: The different control architectures differ in the number of concurrent controllers. In the third experiment, we want to test how changing the overall capacity of the whole architecture influences performance. On the one hand, we want to exclude that decentralized controller gain an advantage due to a larger capacity as they simply consisted of multiple control modules that were setup using the same simple neural network model. On the other hand, varying neural network size allows us to analyze how robust the different architectures are with respect to selection of hyperparameters.

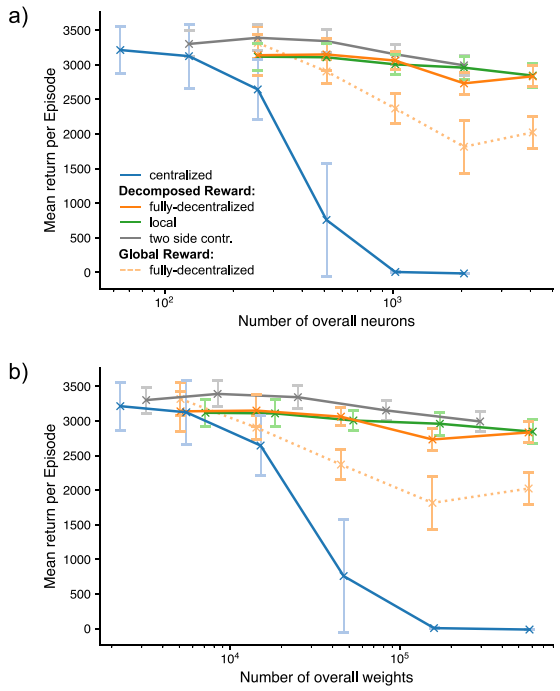

Fig. 11. Comparison of locomotion performance for different sizes of neural networks used in the architectures. Shown is mean performance (y-axis) over ten seeds for each neural network configuration at the end of training (after 20 million simulation steps). Horizontal axis represents size of networks, i.e. overall number of neurons (shown in a) in all networks of that particular architecture (each controller consists of a policy and a value function network with each two hidden layers; in the decentralized case there are four (or two) such controllers in an architecture, leading to more neurons). In (b) the same data is plotted, but overall number of parameters of controller is used for size comparison.

Experimental procedure: We varied the size of the used neural network models for the different architectures and compared the performance for the trained controller with respect to the overall representational capacity of that particular architecture. In order to compare this capacity, we use both, number of neurons and number of weights.

Each controller is represented as a neural network that—using PPO as an actor–critic approach, see Section 4.1—consists of a policy network and a network for approximating the value function (details on neural networks, see 4.3). Both networks share the input space, but the action network provides for each action dimension two outputs (as this produces a stochastic policy, one output represents the mean and the other the standard deviation for that particular degree of freedom), and the value function network provides a single value estimate. Both networks consist of two hidden layers. In all the previous experiments, we used 64 neurons for each hidden layer. In comparing performance based on the capacity of the model, we varied systematically the size of the hidden layer (detailed results and network sizes are given in the supplement, Table S6). Sizes for the different architectures were chosen in a way to use comparable overall number of neurons and weights. We selected smaller and larger sizes. For each size, 10 seeds were trained on flat terrain, and the mean performance at the end of training was used to compare the performance.

Result — Influence of neural network size: The results for five architectures are shown in Fig. 11. First, for the baseline approach (the centralized approach), we observe a known and expected problem in overfitting, as the performance drops considerably for larger architectures. Further, we can see that the initially chosen size of 64 neurons per hidden layer (third datapoint from left in all graphs) appears as a reasonable selection. This is in agreement with the literature on these types of locomotion problems (Andrychowicz et al., 2020). In contrast, for the decentralized approaches there is nearly no drop off in performance for any size when using the decomposed reward function for guiding learning. As one further variation, we considered for the fully decentralized architecture again learning towards a global reward (orange dotted line in Fig. 11) as done in experiment one. Using the global reward had a negative impact on overall performance, but only for larger neural networks. Importantly, there is only a slight drop of performance when compared to the centralized architecture which was also trained using a global reward, but basically produced no positive return at all for larger network sizes and showed severe overfitting. This is an agreement with our assumption that a decomposed reward structure mostly helps facilitate learning and leads to faster learning.

Summary: Considering the neural network size as one influential parameter that differs between the different architectures, we found that a specific selection of network size was only important for the standard centralized approach. The decentralized approaches showed robust to variations of neural network size. Furthermore, for other hyperparameters we selected values and tuned values explicitly on the centralized approach and simply applied these on the decentralized approaches as well which did not appeared negatively impacted. The decentralized architectures showed quite robust to hyperparameter selection.

5.4. Experiment 4 — Learning to walk at a given target speed in uneven terrain

Overall goal of experiment 4: The goal of this task is to turn towards a more realistic scenario which brings us closer to realization on a robot. This will integrate what we have learned in the previous experiments. We aim for training an agent towards a stable walking speed in increasingly more and more difficult terrain. Our hypothesis is that training in more diverse terrains leads to more robust controller under varying conditions.

Experimental procedure: We trained four architectures that were selected as they performed well in the previous experiments and cover the whole range of decentralization and information scope. The setting was changed for these training runs. First, the agent was explicitly trained in a curriculum of uneven terrain that got over time more and more difficult. The maximum difficulty—due to the terrain—was increased during the first ten million simulation steps linearly (smoothness ranges from initially 1.0, equaling flat terrain, to 0.8 which corresponds to uneven terrain). Height fields were randomly generated for every simulation episode with a smoothness value randomly chosen from the interval of minimal smoothness (maximum difficulty) to flat terrain. This value was used to randomly generate a terrain. As a consequence, after the first half of training the controller were still facing easy environments in flat terrain and quite challenging terrains.

Secondly, we were not aiming for a maximum velocity as widely used as a benchmark. For the long term goal of application on a real system, we would not necessarily be interested to run as fast as possible, but would aim more for well coordinated behavior which, for example, is not too taxing for the servo motors or joints (see Azayev & Zimmerman, 2020 and discussion in Section 4.2). Therefore, we aim for specific target velocities (reward see Table 2) which is nowadays applied more and more in locomotion tasks. Reaching that target velocity was rewarded. We, furthermore, chose the decomposed reward structure for the decentralized architectures as the local reward structure had a positive impact on training and performance (experiment 1.3, Section 5.1.3). Two different target velocities were used randomly

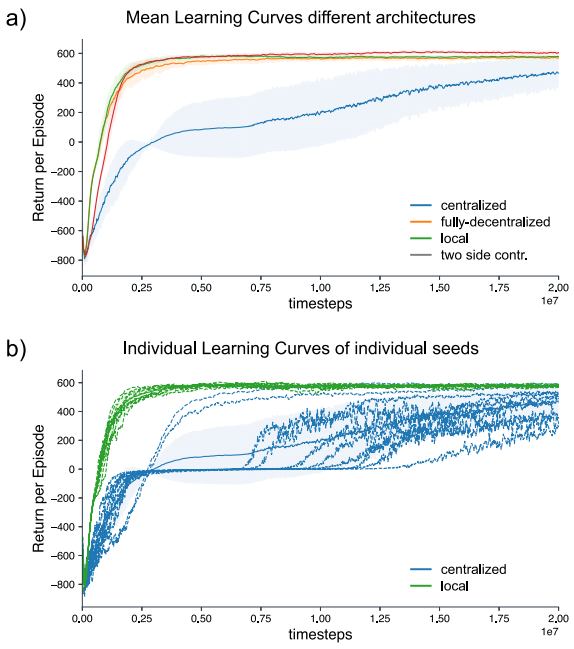

Fig. 12. Learning curves for different controller architectures over learning time in experiment 3, trained to reach two target velocities on increasingly more uneven terrain: (a) Mean return per episode—calculated over 10 seeds—for all different learning architectures is shown during learning, given in simulation steps (interactions with the environment). Standard deviation between seeds is given as shaded area. (b) Individual learning curves over learning time for centralized (blue) and decentralized (green) controller architecture with local information from both neighboring legs, given in simulation steps (interactions with the environment) and shown is the return per episode. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

during training that would correspond to slow and fast walking in uneven terrain (target velocity of 2.0 $\mathrm{m\,s^{-1}}$ or 1.0 $\mathrm{m\,s^{-1}}$). These were randomly selected for every simulation run. Target velocity was used as an additional input to the control network(s), and we trained four selected architectures (for overview of all controllers, see Table 1). We again used a centralized approach as a baseline which is compared to, first, a fully decentralized approach of four controllers with only information from the controlled leg. Secondly, a decentralized controller with information from the two neighboring legs. Last, the architecture consisting of two controllers, one for each side (left, right).

5.4.1. Research Question 4.1: Can decentralized controller learn coordinated walking at a given target speed on the same level as a centralized approach, and do we observe a gain in learning speed?

Result — Learning of a target velocity in uneven terrain: All controller were able to learn locomotion at a given target velocity during 20 million simulation interaction steps (Fig. 12, and see supplemental video S2). Learning progressed differently for the different control architectures, as observed in experiment 1 (Section 5.1). For the centralized approach, learning developed in two phases. Initially, it took the centralized architecture quite a long time to reduce the high costs in the first part of learning (for mean values in Fig. 12(a) up to around three million steps). As a result, all actions were inhibited. Only afterwards, in a second phase, learning of movements towards gaining rewards is observable. All decentralized control architectures show smooth and much faster learning without getting stuck in the local minima around zero. Considering individual learning runs (Fig. 12 b) even the best centralized learning seed required three to four time of learning until reaching a similar plateau of learning compared to the worst local decentralized approach (importantly, all these approaches are quite close together).

Analysis of emergent gaits and produced footfall patterns is shown in the supplement, section S-VII. The qualitative view of example footfall patterns indicates anti-phase relationships between neighboring legs which was in particular pronounced for hind legs. Even in the case of the decentralized architectures in which no explicit communication exists between those hind legs.

Summary: Learning differed between centralized and decentralized architectures. Decentralized architectures quickly learned to walk at different target velocities and in uneven terrain. The centralized architecture learned much slower and often got stuck for a long time during which the controller only avoided incurring any costs and did not move.

5.4.2. Research Question 4.2: How does the different controller generalize to variation of terrain? I.e. what is the influence of decentralization and locality of information on behavior generalization performance?

Goal: Again, we want to compare the capabilities of the different architectures (see Section 5.2). On the one hand, with respect to our behavioral measurements, on the other hand, concerning generalization to more challenging, bumpy terrain.

Approach: We evaluated the different trained control architectures on terrain of increasing difficulty. We systematically varied the smoothness of the surface from a smooth surface (1.0 which equals flat terrain) towards more bumpy surfaces (smoothness factor of 0.9, 0.8, 0.7, 0.6). On the one hand, we evaluated control architectures on uneven terrain which they faced already during learning (smoothness in range of 1.0 to 0.8). On the other hand, we tested for generalization on more difficult terrain (smoothness of 0.7 and 0.6). In all cases, target velocity was set to the two velocities used during training. For each controller architecture, each of the 10 different trained seeds was tested for 100 simulation runs of 1000 control steps each (equaling 50 s).

Result — Performance on uneven terrain: Mean results for the different controller architectures at high velocity are shown in Fig. 13 (for slow walking, see supplemental, Fig. S4). As the costs are going up on more uneven terrain, we, first, observe an expected drop in overall return with increased difficulty. Furthermore, the centralized approach performs significantly worse on all terrains for fast walking.

Secondly, again, the different control architectures show different generalization capabilities. While all architectures show a considerable drop for novel terrain, it appears that integrating more information is helpful for more difficult terrains. The fully decentralized approach and the two-side controller show a much larger drop in performance compared to the decentralized local approach. All these decentralized approaches started on a similar performance level on known terrain as used during training (Fig. 13(a)). On the most difficult novel terrain (smoothness 0.6), we found that the decentralized local approach performed much better compared to the other approaches (Fig. 13(b)). We used a Kruskal–Wallis test that showed a significant, very strong effect on how well the controller performed on novel terrain when compared for architecture (H statistic $= 17.327$, 4 groups, each 10 values, $p < .001$, $\epsilon^2 = 0.444$, $\eta^2 = 0.398$ which indicates the portion of variance in the dependent variable that is explained by the independent variable). The post-hoc test using Dunn's test with Bonferroni correction showed significant differences: The decentralized local architecture showed a highly significant ($p < 0.01$) difference in performance on novel, difficult bumpy terrain compared to the fully decentralized and the centralized architecture.

It is important to note that the return values are not comparable to the return values from previous experiments. The return

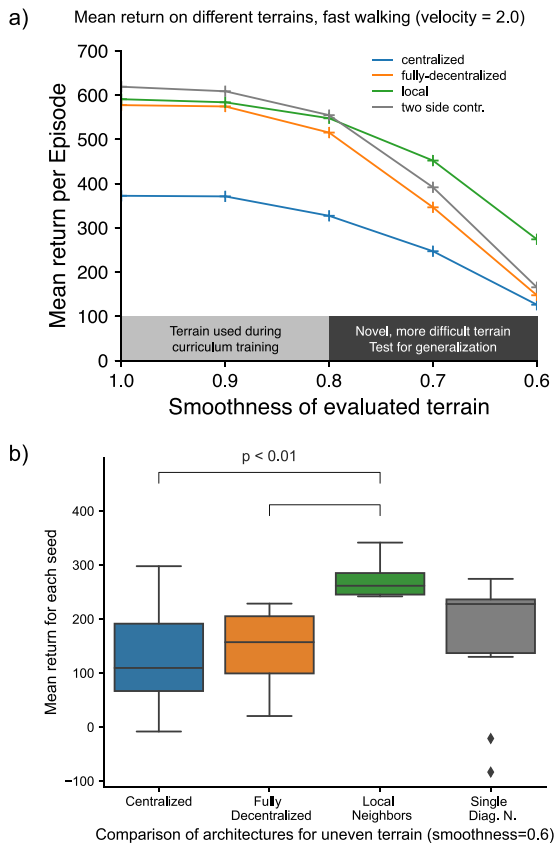

Fig. 13. Evaluation on different types of terrain: For each architecture, all trained controller are evaluated for 100 runs on more and more uneven terrain (smoothness 1.0 is flat terrain and 0. would be very bumpy terrain). In (a) mean return for the four architectures is shown and how performance decreases for more difficult terrain. Target velocity is set to 2.0 (fast walking). Panel (b) compares mean returns for bumpy terrain (smoothness = 0.6) for the different architectures. Local architecture performs highly significant better compared to fully decentralized and centralized architecture.

is composed of a reward for reaching target velocity (linear scale, with maximum reward of 1.0 when moving at target velocity; therefore, maximum return over 1000 control steps would be 1000; see Table 2). On the other hand, there are as well costs factored in: A control cost and a contact cost that initially dominate the overall return and combine early-on to costs of around 800. Costs got considerably reduced during learning, even for more difficult terrain (overall costs for the centralized approach were at around 210 and around 180 for the decentralized approach that used local information at the end of training).

Result – Efficiency of learned locomotion: As we are interested in efficiency of resulting locomotion, we again used 100 evaluation runs of each 1000 control steps for each of the ten different seeds for each of the four different control architectures. Evaluation was run on different types of terrain. Efficiency is again measured as cost of transport (see Section 5.2.2) for the resulting walking behavior. These CoT values are better comparable as they are with respect to the same target velocity which has an influence of possible efficiency (see Table 5).

With respect to the cost of transport, we again compared the results using Kruskal–Wallis test (Kruskal & Wallis, 1952) in all different conditions (variation of velocity and unevenness of terrain) and found a very strong effect of architecture (H statistic > 21.92 for all conditions, 4 groups, each 10 values, $p < .001$, $\epsilon^2 > 0.56$, $\eta^2 > 0.52$). Using Dunn's test with Bonferroni correction as a post-hoc test showed that the decentralized local controller has highly significant lower costs of transport compared to the centralized approach ($p < .01$) for all conditions. The fully decentralized architecture produced significantly more efficient gaits compared to the centralized approach for known terrain (flat or smoothness of 0.8) for high velocity. Again, we find here a trend that for more difficult tasks decentralization provides an advantage.

Table 5
Evaluation of different control architectures. Each architecture was trained ten times (on a curriculum of terrains, reward aims for reaching a specified velocity) and each of these seeds was evaluated for 100 episodes on multiple different terrains (flat terrain; uneven terrain, smoothness 0.8; bumpy terrain, smoothness 0.6). Given is the mean return and cost of transport over all episodes per architecture for a specific terrain type and the selected target velocity.

Configuration	Low velocity (1.0)		High velocity (2.0)	
	Return	CoT	Return	CoT
Flat terrain (used during training)				
Centralized	547.0	9.911	372.4	7.438
Fully decentralized	588.0	8.072	577.4	5.665
Decentr., local inf.	585.7	7.277	590.8	5.442
Split c. (sides)	621.5	7.958	619.0	5.699
Uneven terrain (smoothness 0.8; used during training)				
Centralized	499.6	10.176	327.1	7.871
Fully decentralized	552.9	8.297	515.3	6.014
Decentr., local inf.	560.3	7.395	547.6	5.689
Split c. (sides)	590.5	8.144	554.7	6.076
Bumpy terrain (smoothness 0.6; unseen, test for generalization)				
Centralized	307.5	11.828	126.1	9.777
Fully decentralized	268.6	9.939	147.5	7.971
Decentr., local inf.	342.8	8.271	274.0	6.686
Split c. (sides)	351.4	9.558	165.5	7.938

Summary: The experiment confirmed results from the first experiment. With respect to the influence of additional information, the earlier experiments brought us to the assumption that additional information becomes more important for increasingly more difficult tasks. The evaluation results support this assumption that generalization benefits from additional information, but only up to a certain degree. As mentioned, the decentralized control approaches performed significant better and at lower costs of transport compared to the centralized approach. The local approach (decentralized, using local information from both neighboring legs) further showed significantly more efficient on unseen, quite difficult terrain when compared to the fully decentralized approach and the two controller approach (one for each side) for high velocity (and for low velocity w.r.t. the fully decentralized approach).

5.4.3. Research Question 4.3: How does the different controller generalize to variation of target velocity?

Goal: Temporal coordination of locomotion appears better characterized as free gaits in which temporal relations emerge from the interaction with the environment (Bidaye et al., 2018; DeAngelis et al., 2019). As locomotion in challenging environments requires such constantly changing adaptations of velocity, we are interested in how the different architectures deal with a continuous variation of velocity.

Approach: During training, two different target velocities (1.0 and 2.0) were used which were randomly chosen for each episode and which were given to the controller as an additional input. In the previous evaluation, we used these two training velocities. As a next step, we are now considering variations of the target velocity. For each architecture, all trained controller were evaluated for 10 runs for target velocities in the range of [0.5, 2.5] with increments of 0.1. This was done again for the different terrains, ranging from flat to bumpy terrain. Overall, each controller was tested for 26 different target velocities on 5 different terrain types 10 times.

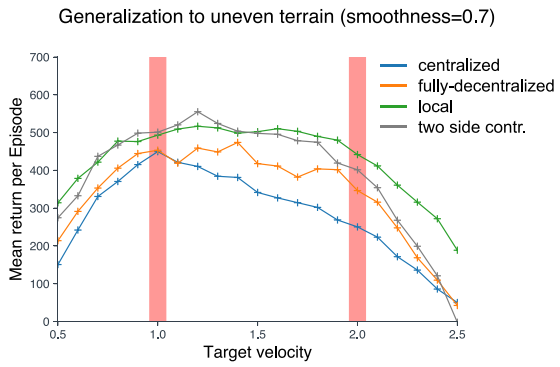

Fig. 14. Evaluation of different target velocities on uneven terrain: For each architecture, all trained controller are evaluated for 10 runs at different target velocities (x-axis, ranging from 0.5 to 2.5 in increments of 0.1). Shown is the mean return for each target architecture (y-axis). Red shaded areas indicate the two target velocities used during training.

Result — Generalization to novel target velocities: Fig. 14 shows the results of the evaluation for a quite uneven terrain (smoothness = 0.7). This terrain presents a hard task that was not used during training (for results on other terrains see supporting information, Fig. S5). The two target velocities used during training are shown as the red shaded area. In between, we observe the mean return as a performance measure when interpolating target velocity between two known values. The decentralized local controller shows smooth interpolation at a high level. This is similar for the two-side controller that shows a single outlier that may be due to the lower number of evaluation runs. The fully decentralized approach shows more variation. The centralized approach shows linear interpolation, but overall performance degrades considerably for higher velocities. Considering extrapolation of target velocities, we see a performance drop for all architectures. This is to be expected for higher velocities in particular, as the simulated robot reaches its limit of possible target velocity. For extrapolation towards lower velocities it is important to note that the calculated reward gets more sensitive towards deviation from the target velocity as it is linearly scaled. Comparing the different architectures, their performance is on a similar level for target velocity 1.0. But for lower velocities this gap appears to widen in favor of the local decentralized controller.

For a better comparison of the architectures, we further show in Fig. 15 how performance of architectures is impacted by variation of terrain and velocity. It is based on the same evaluation runs as before, but now given in a box plot that, in addition, visualizes the variance of each approach. First, we consider walking at a target velocity of 1.0 as used during training. There is only little impact when switching from flat terrain (Fig. 15(a), left) towards uneven terrain (middle) and mainly the centralized global approach is affected. For bumpy terrain (on the right), we observe that, for the low target velocities, performance goes down for all architectures (see as well Fig. S3). All architectures appear to generalize similarly when the task is not too demanding (we showed before that for the high velocity there is not only a highly significant difference, but that the gap also widens between the local and the fully decentralized as well as centralized architecture, Fig. 10).

When choosing a novel target velocity—in between the two used for training—, we see a similar effect (Fig. 15(b)). For flat terrain, mostly the variance of the gained returns of the centralized architecture increased. Switching to uneven terrain does not hurt the performance too much, until we get towards bumpy terrain. For bumpy terrain (shown on the right), all controllers are affected. But again, the fully decentralized approach is affected the most. It appears—in agreement with the other experiments—that for more difficult terrain, more information is advantageous. The last row (Fig. 15(c)) shows selection of a very slow target velocity outside the range of the two target velocities during training (extrapolation to slow walking). As detailed above (see Fig. 14), all architectures perform considerably worse. But switching to uneven terrain does not impact controller performance. Only when further increasing difficulty, we observe a drop in performance. This time the margin between, on the one side, local decentralized (mean return: 228.2) as well as the two-side architectures (mean return: 218.8) and, on the other hand, the centralized (mean return: 113.1) as well as fully decentralized architecture (mean return: 111.1) widens. This further confirms our previous observations that, for a decentralized approach, additional information helps when facing more demanding and, to a certain extend, even novel tasks. But that full global information does not appear to further help in these cases.

Summary: For generalization towards target velocities to values not used during training, we see that the decentralized approach using local information from both neighboring legs performs relatively well across different types of terrain. Interpolation between used velocities is realized on a high level, while for extrapolation towards even slower or higher velocities we see a drop in performance (as for all the architectures).

5.5. Experiment 5: Analysis of importance of input features

Research Question 5: Which input information is used by a trained standard centralized controller? Does the learned importance of input features on action selection align with our assumed and exploited organization?

Goal: As we systematically varied the scope of input information and the selection of input features, we, at last, investigate which input features influence control and to what extend. Therefore, we want to measure the importance of each feature dimension in a well performing controller that in principle has access to all available information.

Approach: We access the importance of an observed feature as the effect on the control signal when applying a small change in a particular input dimension. This is comparable to saliency map based approaches which highlight features in input space that contribute to decisions of neural networks (Simonyan, Vedaldi, & Zisserman, 2013). Most of this work on explaining the function of neural networks has been traditionally in the area of visual—and in particular classification—tasks (Alber et al., 2019; Zeiler & Fergus, 2014). We are following here a similar approach in that we are evaluating how the output of the network changes when altering the input, i.e. we algorithmically compute the gradient of the output with respect to a given input which is a type of input modification method (Grün, Rupprecht, Navab, & Tombari, 2016).

As we are dealing with DRL, we are accumulating this gradient over multiple rollouts and runs of the agent in the environment. We selected the centralized architecture that uses global information for the analysis. In this architecture, all information for control is available for the decision making for each action dimension. After successful training, the control network should have learned which information is necessary and should accordingly weight these input information higher. In contrast, for input channels that do not contain any meaningful information the influence would be narrowed. In principle, input could contain all available (44) feature dimensions, and we want to understand which dimensions are used to what extend. Each seed from experiment four (Section 5.4) of the centralized architecture was run for 10 evaluation runs (producing multiple trajectories τ) of each $t = 1000$ control steps. We are interested in how a change in the input (observation o_i over all $i = 44$ dimensions) changes the

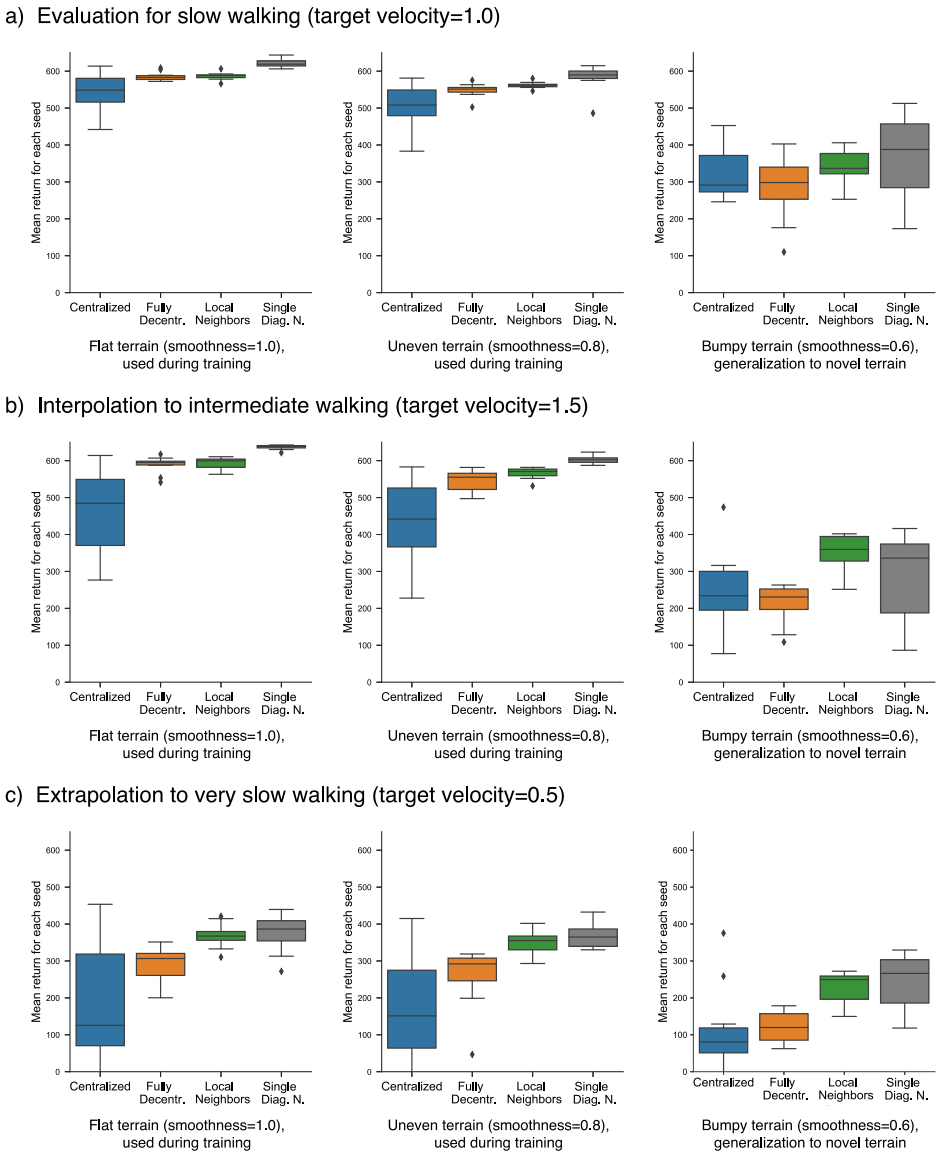

Fig. 15. Comparison of four different control architectures at different target velocities on different terrains. For each architecture, all ten trained controller are evaluated for 10 runs at different target velocities and for a selected terrain smoothness. Shown is the mean performance and standard deviation for each architecture. Left column always shows evaluation on flat terrain, middle column on uneven terrain (which was as well used during training), right column shows generalization to bumpy terrain. In (a) mean returns for a low target velocity (1.0, used during training) are shown. In (b) an intermediate target velocity was selected (1.5, right between the two target velocities used during training). Last, (c) a very low target velocity was selected (0.5).

output of the network (μ_j – mean value of the normal distribution from which actions are sampled):

$$\text{Importance Map } M_{ij} = \sigma_j \sum_{\tau} \sum_{t} abs\left(\frac{\partial \mu_j}{\partial o_i}\right)$$

For approximating this gradient, we measured the original output of the policy network during each control step and computed the gradient numerically with respect to the individual input dimensions of the observation space. As the different input dimensions are of quite different scale and variation during walking, we had to normalize the computed gradient for which we used the standard deviation of the particular feature dimension (measured over time as σ_i). As a result, we get for each of the 44 input dimensions how it affects the eight control signals (Fig. 16(a)).

Result — Importance of input features: There were ten trained centralized controllers (used from experiment four, trained on uneven terrain and for two different target velocities) which did not show a large difference w.r.t. the computed importance maps. Importance maps did not differ for different tasks (different velocity, uneven terrain) too much as well. Therefore, we only provide information for one well performing controller. There are two main observations: First, there are two stair-like structures visible in the joint angle and joint velocity inputs. These high diagonal importance values indicate that for each joint control signal, information from that particular joint are crucial and highly important. To further analyze the impact of local information, we grouped the 44 inputs depending on the scope of information into five groups: Global information (about torso), local information from the controlled joint, local information from that leg (i.e. from the other joint in that leg), information from the two neighboring legs, and information from the diagonal leg (Fig. 16(b)). Comparing this to a uniform distribution of importance (as we would get for an untrained network), we can see that importance of global information is overweighted. The same is true for information from the single controlled leg for which importance is

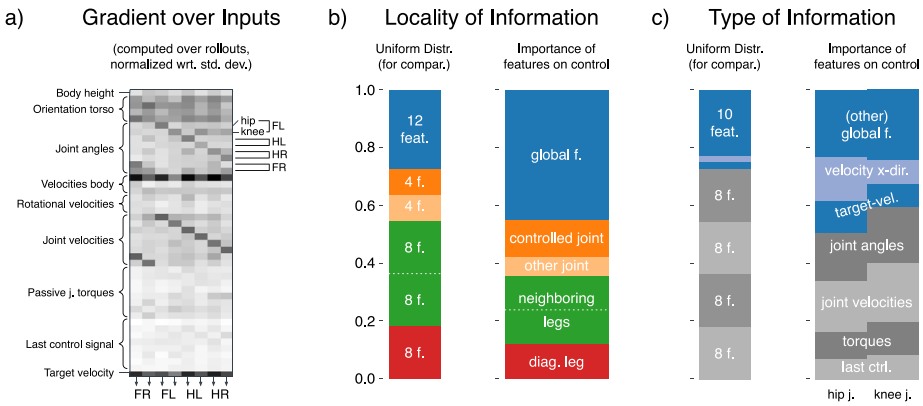

Fig. 16. Importance of input features for selecting control signals: (a) Importance map showing how a small change in one input (rows, *y*-axis, named on the left) feature dimension affects the control signals (columns, shown on the bottom). For the joint inputs, ordering is front left leg (hip, followed by knee joint for all legs), hind left leg, hind right leg, and front right leg. (b) Importance grouped for locality of information is shown on the right (as there is no difference for different legs and joints, these are integrated as well): Global information (body height, torso orientation, torso velocities, target velocity), local information from the controlled joint, information from the second joint of the same leg, information from all four joints of the two neighboring legs, and information from the two joints of the diagonal leg. On the left, an assumed uniform distribution simply based on the number of features for that group is given for comparison. (c) Importance is grouped by type of information: Global information (body height, torso orientation, torso velocities without x velocity which showed to be very important), velocity in *x*-direction (single dimension), target velocity (single dimension), joint angles, joint velocities, passive torques, and last control signals. On the left, an assumed uniform distribution simply based on the number of features for that group is given for comparison.

twice as high as assumed by a uniform distribution. All the other information are equally weighted (w.r.t. to the number of input channels).

Secondly, we observe that different types of input appear to be of different importance. In particular, target velocity (bottom of 16(a)) and velocity in *x*-direction (first dark line of body velocities in 16(a)) appear prominent in the importance map. Therefore, we grouped the importance map with respect to different types, differentiating seven groups: Global features (not including target velocity and velocity in *x*-direction), velocity in *x*-direction, target velocity, joint angles, joint velocities, passive torques measured in each joint, and, last, control signal (Fig. 16(c)). As a comparison, we use an assumed uniform distribution (shown on the left). Further, we distinguished the two joints of each leg, hip (which might be misleadingly named, but this produces the for- and backward movements) and knee joint (we summed these up for all legs as there was no difference between legs). As a result, we found that passive torque and last control signals appear as of reduced importance. The rest of the global features, joint angle, and velocity information appear of expected importance. But target velocity and velocity in *x*-direction are of a largely increased importance. This importance reflects that the task is to keep track of the provided target velocity. This is even more pronounced in the hip joint which appears sensible as this joint is contributing the most to forward movement of the body.

Summary: In a trained controller that has access to all available information, we found that, first, global information on current velocity and target velocity as well as body orientation is quite important. Secondly, local information from the controlled joint itself is crucial and highly important for control, while other joints show a much lower influence.

6. Discussion and conclusion

The central research question in this work is on the impact of decentralization of control architectures, the scope of input information, and the structure of the reward function. Following biological inspiration, we considered different types of decentralized and modular control structures which lead to local and concurrent processing of control. We used locomotion of a four-legged simulated robot in uneven and unknown terrain as a task as it requires adaptive behavior, i.e., adapting to unpredictable challenges in the environment immediately. In particular, we considered variation of three characteristics: First, the degree of decentralization of control which was systematically varied for the chosen four-legged robot. Second, this had a further influence, as we were employing a Deep Reinforcement Learning approach, on the reward and cost structure that drives learning. We considered decomposition of the reward following the degree of decentralization as we were differentiating local costs. Third, the input scope of the specific controllers, ranging from small input spaces of local observations to larger input spaces for each controller including all information. As we considered these characteristics, we wanted to, first, answer what kind of information is required for control of locomotion and how this affects learning. And, secondly, how this translates to generalization performance in unseen environments (for an overview of results, see Table 6).

First, for learning locomotion we found that decentralized approaches—that only have access to limited information—are capable of learning well performing behaviors. This corroborates our earlier qualitative finding showing this for a simulated six-legged robot (Konen, Korthals, Melnik, & Schilling, 2019; Schilling et al., 2020) that was able to learn running as fast as possible using only a decentralized architecture. While this previous work already indicated that learning in decentralized architectures might progress faster, our current data do not only show this as a much more prominent effect, but allow further conclusions that significantly go beyond that: In contrast to the previous study, we considered a more refined reward strategy during learning which also accounted for movement costs. This decomposition of costs as part of the reward showed to have a large effect on learning speed. Local costs helped to drive learning considerably faster towards high performing control. This is in agreement with assumptions on reward decomposition which is known to facilitate faster learning (Schneider et al., 1999), but usually restricted to simpler problems. While recent work has put a focus on reward decomposition, we think that for articulated robots such a decomposition of rewards is intrinsically given through the morphology and should be exploited. In general, it appears that a modular control structure allows to exploit local cost structures which can help DRL considerably. Usually, during exploration in the context of DRL, high dimensional actions are stochastically tried out and there is only a single reward that cannot be disentangled. In contrast, in a decentralized approach

Table 6
Summary of selected results for comparison of different controller types: Shown are main results from the experiments for the different control architectures. The architectures are, first, differentiated with respect to control (action space) into centralized and decentralized approaches. On a second level, we can further distinguish—for the decentralized architectures—what kind of information (state space) each controller is using as an input. Rows show different experiments and evaluations. Single cell colors provide a relative comparison of results for a single measurement: For each row, a mean value is calculated over all different approaches. The color of a cell is set to white if the value is close to the mean; it is set to a green color if the value is better (higher for velocities and returns, lower for costs) compared to the mean, and to red if it is worse. Saturation of color indicates deviation of the mean for that particular cell (fully saturated color represents a deviation of more than one standard deviation of the mean).

	Control architecture (action space variation)	Centralized	↔	2 Controller		↔	Decentralized, 4 controller				
	Input scope (observation space var.)	Global		Each side	diag. legs		Both neigh.	Single n.	diag. leg	tow. front	ctrl. leg
Experiment 1: Effect on decentralization, locality of information, reward decomposition (Section 5.1)											
1.2	Learning performance (end of training), Section 5.1.2										
	Mean Learning Performance	1251.83		**2706.61**	2353.34		2503.28	2510.03	2486.14	2542.02	2543.97
Experiment 2: Generalization to uneven terrain and efficiency (Section 5.2)											
2.1	Generalization to uneven terrain (smoothn. 0.8), Section 5.2.1										
	Velocity $m\,s^{-1}$	2.55		**2.60**	2.43		2.55	2.37	2.37	2.32	2.15
	Cost of Transport	9.324		8.391	8.467		**7.283**	7.967	7.420	7.594	7.424
2.2	Performance of trained controller (after 20M steps) on flat terrain (from exp. 1) Section 5.2.2										
	Velocity $m\,s^{-1}$	3.57		**3.85**	3.51		3.54	3.48	3.47	3.43	3.33
	Cost of Transport	8.224		6.758	7.021		6.169	6.244	6.137	**6.056**	6.283
Experiment 4: Walking at a given target velocity (Section 5.4)											
4.1	Performance of trained controller (uneven terrain, used in curriculum tr.), Section 5.4.1										
	Mean return, low velocity	499.6		**590.5**	/		560.3	/	/	/	552.9
	Mean return, high velocity	327.1		**554.7**	/		547.6	/	/	/	515.3
4.2	Generalization trained controller (bumpy terrain), Section 5.4.2										
	Mean return, low velocity	307.5		**351.4**	/		342.8	/	/	/	268.6
	Mean return, high velocity	126.1		165.5	/		**274.0**	/	/	/	147.5
	Efficiency of trained controller (uneven terrain, used in curriculum tr.), Section 5.4.2										
	Cost of transport, low velocity	10.176		8.144	/		**7.395**	/	/	/	8.297
	Cost of transport, high velocity	7.871		6.076	/		**5.689**	/	/	/	6.014

we exploit the given local cost structure to which there is direct access. This helps to avoid costly local actions, unless they show a positive net reward for the whole system.

The second main contributing factor for faster learning appears to be the smaller search space when only relying on local sensory information and the resulting lower dimensional state space. Leaving out distant sensory signals does not appear to negatively affect the learned controller for the tasks of dealing with known uneven terrains. This information on the contributions of other more distal legs might be redundant as, for example, lifting of another leg has direct consequences that can be sensed in a leg without requiring direct neural transmission of this information. Such an approach has been demonstrated in robots in Owaki and Ishiguro (2017). Furthermore, in the last experimental section we used a visualization approach from explainable AI that highlighted the importance of local information in a centralized controller that had access to all information during training. The centralized controller learned to mostly rely on information from the controlled joint. While we considered state spaces following the morphology of the robot, we want to turn in the future towards uncovering meaningful state spaces, e.g. using graph neural network based approaches, comparable to the approach in Huang et al. (2020) who use a message passing approach for sharing information across body morphology or to Sanchez-Gonzalez et al. (2018) who are further incorporating and learning dynamics across a graph for planning of movements.

When turning towards generalization towards more challenging terrains then experienced during training, all considered architectures were, expectedly, affected by the difficulty. But to a different degree: A fully decentralized approach performed worst in such a scenario. We found that including more information from other legs appeared advantageous and lead to higher returns. However, it appeared that it is not necessary to include information from all legs. In fact, the local controller that used information from only two neighbors performed best in novel and more unpredictable terrain. This increased importance of sensory inputs when facing a more difficult locomotion task is again well in agreement with research on locomotion in insects that points out that, for slower walking and climbing, control is more and more sensory driven (Neveln, Tirumalai, & Sponberg, 2019; Schilling, Hoinville et al., 2013) which is also assumed for animals in general (Dickinson et al., 2000) and humans (Kuo, 2002). This finding is well in agreement with our handcrafted model Walknet that uses a similar local connection structure on a six-legged robot (Schilling, Hoinville et al., 2013), but is much more limited with respect to agility of motor behavior.

Last, we found that decentralized control architectures showed higher robustness of the learning process. In the third experiment (Section 5.3), we found that the centralized approach was brittle with respect to hyperparameter variation and degraded substantially when changing the hidden layer size of the neural network—it overfitted. In contrast, the decentralized approaches produced well performing solutions without any significant performance drop when varying the hidden layer sizes and was learning more robustly.

Finally, we want to acknowledge that this type of interdisciplinary research appears to benefit the individual disciplines: For the field of machine learning and DRL, we learned that decentralized control architectures can facilitate DRL as they speed up the learning process and can yield high robustness with respect to the selection of hyperparameters related to the specific neural network model. Furthermore, after learning, the controller showed adaptive behavior and an improved ability for generalization. Admittedly, decentralization cannot be applied in all DRL approaches. With the current focus of the last years on game playing or environments using discrete actions, typical environments do not offer an easy factorization or decomposition into modules (but see, for example, Chang, Kaushik, Levine, and Griffiths (2021) who aim for modular decomposition when assigning rewards in a simple RL task). Still, decentralization and decomposition of reward might be worthwhile when dealing with articulated actuators as are legged robots or in manipulation tasks. Finally, our study provides a detailed analysis of the potential benefits of modularization in artificial neural networks (cf. Amer & Maul, 2019), in a concrete application scenario.

Correspondingly, we think that this research strategy has also a lot to offer for biology: Simulated robots and implemented

hypothetical models have always been used as tools to understand general principles and their implementation as mechanisms (not only in motor control) (McClelland, 2009; Webb, 2001). This often requires detailed work in setting up and parametrizing models in small and narrow experimental paradigms. Learning-based approaches not only allow broader applications in more diverse environments, but also allow to observe the process of finding solutions. Based on such observations, one can compare different mechanisms or variations of mechanisms that can directly translate back to insight or possible experimentation in biology. In the present case of control structures for walking, we started with one organizational principle of motor control, decentralization, and in particular pointed out detailed research on decentralization in insects. This work comprises behavioral analysis and neuroscientific studies. One still open and hotly debated question is how local ensembles of neurons and higher levels interact (see, for example, recently published results in Feng et al. (2020) on how lower level control and top-down commands interact in backward walking of *Drosophila*). Our computational study provides interesting complementary insights on local and decentralized computation. Local control is often seen as a necessity, i.e., due to sensory delays it would be too slow to rely on a central controller (a brain) to react. In such a view, decentralized control based on local information appears only as a required compromise. But—as we are not even considering latencies that would benefit the local approaches—our results show that decentralized control is a viable approach that actually simplifies the problem at hand and can compete on the same level as a centralized approach. Even further, it indicates a tradeoff for information: A decentralized approach that only had information to information from neighboring legs showed best generalization performance towards more unpredictable environments. Importantly, we still found that some global (top-down) information is crucial and required for control. From our point of view, decentralization and limited scope of information offer a way to cut up the large sensory state and control space into well manageable subspaces. Such a view on different concurrent control processes differs from the standard paradigm of DRL in which decision making is assumed in one central control unit. From our point of view, it is worthwhile to further consider such a modular perspective for Deep Reinforcement Learning and neural networks.

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 A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.neunet.2021.09.017.

References

Ache, J. M., & Matheson, T. (2013). Passive joint forces are tuned to limb use in insects and drive movements without motor activity. *Current Biology*, *23*(15), 1418–1426. http://dx.doi.org/10.1016/j.cub.2013.06.024, URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3739007/.

Akay, T., Ludwar, B. C., Göritz, M. L., Schmitz, J., & Büschges, A. (2007). Segment specificity of load signal processing depends on walking direction in the stick insect leg muscle control system. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, *27*(12), 3285–3294. http://dx.doi.org/10.1523/JNEUROSCI.5202-06.2007.

Alber, M., Lapuschkin, S., Seegerer, P., Hägele, M., Schütt, K. T., Montavon, G., et al. (2019). Innvestigate neural networks!. *Journal of Machine Learning Research*, *20*(93), 1–8, URL: http://jmlr.org/papers/v20/18-540.html.

Alon, U. (2006). *Chapman & Hall/CRC mathematical and computational biology*, *An introduction to systems biology: Design principles of biological circuits*. Taylor & Francis, URL: https://books.google.de/books?id=tcxCkIxzCO4C.

Amer, M., & Maul, T. (2019). A review of modularization techniques in artificial neural networks. *Artificial Intelligence Review*, *52*(1), 527–561. http://dx.doi.org/10.1007/s10462-019-09706-7.

Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., et al. (2020). What matters in on-policy reinforcement learning? A large-scale empirical study. arXiv:2006.05990 [cs, stat]. arXiv:2006.05990.

Arber, S., & Costa, R. M. (2018). Connecting neuronal circuits for movement. *Science*, *360*(6396), 1403–1404. http://dx.doi.org/10.1126/science.aat5994, URL: http://www.sciencemag.org/lookup/doi/10.1126/science.aat5994.

Arena, P., Patanè, L., & Taffara, S. (2021). Energy efficiency of a quadruped robot with neuro-inspired control in complex environments. *Energies*, *14*(2), http://dx.doi.org/10.3390/en14020433, URL: https://www.mdpi.com/1996-1073/14/2/433.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). A brief survey of deep reinforcement learning. *IEEE Signal Processing Magazine*, *34*(6), 26–38. http://dx.doi.org/10.1109/MSP.2017.2743240, URL: http://arxiv.org/abs/1708.05866.

Azayev, T., & Zimmerman, K. (2020). Blind hexapod locomotion in complex terrain with gait adaptation using deep reinforcement learning and classification. *Journal of Intelligent and Robotic Systems*, *99*(3–4), 659–671. http://dx.doi.org/10.1007/s10846-020-01162-8.

Beer, R. D. (1990). *Intelligence as adaptive behavior: An experiment in computational neuroethology*. USA: Academic Press Professional, Inc..

Beer, R. D., Chiel, H. J., & Sterling, L. S. (1990). A biological perspective on autonomous agent design. *Robotics and Autonomous Systems*, *6*(1), 169–186. http://dx.doi.org/10.1016/S0921-8890(05)80034-X, URL: http://www.sciencedirect.com/science/article/pii/S092188900580034X. Designing Autonomous Agents.

Bellicoso, C. D., Jenelten, F., Gehring, C., & Hutter, M. (2018). Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots. *IEEE Robotics and Automation Letters*, *3*(3), 2261–2268. http://dx.doi.org/10.1109/LRA.2018.2794620.

Bidaye, S. S., Bockemühl, T., & Büschges, A. (2018). Six-legged walking in insects: how CPGs, peripheral feedback, and descending signals generate coordinated and adaptive motor rhythms. *Journal of Neurophysiology*, *119*(2), 459–475. http://dx.doi.org/10.1152/jn.00658.2017.

Billard, A., & Kragic, D. (2019). Trends and challenges in robot manipulation. *Science*, *364*(6446), eaat8414. http://dx.doi.org/10.1126/science.aat8414, URL: https://science.sciencemag.org/content/364/6446/eaat8414.

Binder, M. D., Hirokawa, N., & Windhorst, U. (2009). Motor control hierarchy. In *Encyclopedia of neuroscience* (p. 2428). Berlin, Heidelberg: Springer, URL: https://doi.org/10.1007/978-3-540-29678-2_3583.

Botvinick, M. M. (2008). Hierarchical models of behavior and prefrontal function. *Trends in Cognitive Sciences*, *12*(5), 201–208.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, *2*(1), 14–23. http://dx.doi.org/10.1109/JRA.1986.1087032.

Brooks, R. A. (1991). Intelligence without reason. In J. Myopoulos, & R. Reiter (Eds.), *Proceedings of the 12th international joint conference on artificial intelligence* (pp. 569–595). San Mateo, CA, USA: Morgan Kaufmann Publishers Inc..

Brown, I. E., & Loeb, G. E. (2000). A reductionist approach to creating and using neuromusculoskeletal models. In *Biomechanics and neural control of posture and movement* (pp. 148–163). Springer.

Carlo, J. D., Wensing, P. M., Katz, B., Bledt, G., & Kim, S. (2018). Dynamic locomotion in the MIT cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 1–9). http://dx.doi.org/10.1109/IROS.2018.8594448.

Chang, M., Kaushik, S., Levine, S., & Griffiths, T. L. (2021). Modularity in reinforcement learning via algorithmic independence in credit assignment. arXiv:2106.14993.

Chatzilygeroudis, K. I., Vassiliades, V., Stulp, F., Calinon, S., & Mouret, J. (2020). A survey on policy search algorithms for learning robot controllers in a handful of trials. *IEEE Transactions on Robotics*, *36*(2), 328–347. http://dx.doi.org/10.1109/TRO.2019.2958211.

Chiel, H. J., & Beer, R. D. (1997). The brain has a body: adaptive behavior emerges from interactions of nervous system, body and environment. *Trends in Neurosciences*, *20*(12), 553–557.

Chiel, H. J., Ting, L. H., Ekeberg, O., & Hartmann, M. J. Z. (2009). The brain in its body: Motor control and sensing in a biomechanical context. *Journal of Neuroscience*, *29*(41), 12807–12814. http://dx.doi.org/10.1523/JNEUROSCI.3338-09.2009, URL: http://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.3338-09.2009.

Clancy, K. B., Orsolic, I., & Mrsic-Flogel, T. D. (2019). Locomotion-dependent remapping of distributed cortical networks. *Nature Neuroscience*, *22*(5), 778. http://dx.doi.org/10.1038/s41593-019-0357-8, URL: https://www.nature.com/articles/s41593-019-0357-8.

Clune, J., Mouret, J.-B., & Lipson, H. (2013). The evolutionary origins of modularity. *Proceedings of the Royal Society B: Biological Sciences*, *280*(1755), Article 20122863. http://dx.doi.org/10.1098/rspb.2012.2863, URL: https://royalsocietypublishing.org/doi/abs/10.1098/rspb.2012.2863. arXiv:https://royalsocietypublishing.org/doi/pdf/10.1098/rspb.2012.2863.

Cruse, H. (1990). What mechanisms coordinate leg movement in walking arthropods? *Trends in Neurosciences*, *13*, 15–21.

Cully, A., Clune, J., Tarapore, D., & Mouret, J.-B. (2015). Robots that can adapt like animals. *Nature*, *521*(7553), 503–507. http://dx.doi.org/10.1038/nature14422.

Dallmann, C. J., Hoinville, T., Dürr, V., & Schmitz, J. (2017). A load-based mechanism for inter-leg coordination in insects. *Proceedings: Biological Sciences*, *284*(1868), http://dx.doi.org/10.1098/rspb.2017.1755.

d'Avella, A., Giese, M., Ivanenko, Y. P., Schack, T., & Flash, T. (2015). Editorial: Modularity in motor control: from muscle synergies to cognitive action representation. *Frontiers in Computational Neuroscience*, *9*, 126. http://dx.doi.org/10.3389/fncom.2015.00126.

DeAngelis, B. D., Zavatone-Veth, J. A., & Clark, D. A. (2019). The manifold structure of limb coordination in walking Drosophila. *ELife*, *8*, Article e46409. http://dx.doi.org/10.7554/eLife.46409, URL: https://elifesciences.org/articles/46409.

Dickinson, M. H., Farley, C. T., Full, R. J., Koehl, M. a. R., Kram, R., & Lehman, S. (2000). How animals move: An integrative view. *Science*, *288*(5463), 100–106. http://dx.doi.org/10.1126/science.288.5463.100, URL: http://science.sciencemag.org/content/288/5463/100.

Dudek, D. M., & Full, R. J. (2006). Passive mechanical properties of legs from running insects. *The Journal of Experimental Biology*, *209*(Pt 8), 1502–1515. http://dx.doi.org/10.1242/jeb.02146.

Dunn, J., & Dunn, O. J. (1961). Multiple comparisons among means. *American Statistical Association*, 52–64.

Dürr, V., Arena, P., Cruse, H., Dallmann, C. J., Drimus, A., Hoinville, T., et al. (2019). Integrative biomimetics of autonomous hexapedal locomotion. *Frontiers in Neurorobotics*, *13*, http://dx.doi.org/10.3389/fnbot.2019.00088, URL: https://www.frontiersin.org/articles/10.3389/fnbot.2019.00088/full.

Dürr, V., Schmitz, J., & Cruse, H. (2004). Behaviour-based modelling of hexapod locomotion: Linking biology and technical application. *Arthropod Structure and Development*, *33*(3), 237–250.

Ellefsen, K. O., Huizinga, J., & Torresen, J. (2020). Guiding neuroevolution with structural objectives. *Evolutionary Computation*, *28*(1), 115–140.

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., et al. (2020). Implementation matters in deep RL: A case study on PPO and TRPO. In *International conference on learning representations*. URL: https://openreview.net/forum?id=r1etN1rtPB.

Feng, K., Sen, R., minegishi, r., Dübbert, M., Bockemühl, T., Büschges, A., et al. (2020). Distributed control of motor circuits for backward walking in drosophila.. *Nature Communications*, *11*, 6166. http://dx.doi.org/10.1038/s41467-020-19936-x.

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning* (pp. 1126–1135). URL: http://proceedings.mlr.press/v70/finn17a.html.

Flash, T., & Hochner, B. (2005). Motor primitives in vertebrates and invertebrates. *Current Opinion in Neurobiology*, *15*(6), 660–666. http://dx.doi.org/10.1016/j.conb.2005.10.011.

Frans, K., Ho, J., Chen, X., Abbeel, P., & Schulman, J. (2018). Meta learning shared hierarchies. In *International conference on learning representations*. URL: https://openreview.net/forum?id=SyX0IeWAW.

Full, R. J., & Tu, M. S. (1991). Mechanics of a rapid running insect: two-, four- and six-legged locomotion. *Journal of Fish Biology*, *156*(1), 215–231, URL: https://jeb.biologists.org/content/156/1/215.

Gabrielli, G. (1950). What price speed? *Mechanical Engineering (ASME)*, *72*(10), 775–781.

Giszter, S. F., Mussa-Ivaldi, F. A., & Bizzi, E. (1993). Convergent force fields organized in the frog's spinal cord. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, *13*(2), 467–491.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics, Vol. 9* (pp. 249–256).

Graziano, M. (2006). The organization of behavioral repertoire in motor cortex. *Annual Review of Neuroscience*, *29*, 105–134. http://dx.doi.org/10.1146/annurev.neuro.29.051605.112924.

Grillner, S. (2003). The motor infrastructure: from ion channels to neuronal networks. *Nature Reviews Neuroscience*, *4*(7), 573–586. http://dx.doi.org/10.1038/nrn1137.

Grün, F., Rupprecht, C., Navab, N., & Tombari, F. (2016). A taxonomy and library for visualizing learned features in convolutional neural networks. In *ICML visualization for deep learning workshop*. URL: http://arxiv.org/abs/1606.07757.

Ha, S., Kim, J., & Yamane, K. (2018). Automated deep reinforcement learning environment for hardware of a modular legged robot. In *2018 15th international conference on ubiquitous robots (UR)* (pp. 348–354). IEEE.

Ha, S., Xu, P., Tan, Z., Levine, S., & Tan, J. (2020). Learning to walk in the real world with minimal human effort. arXiv:2002.08550 [cs]. arXiv:2002.08550.

Hart, C. B., & Giszter, S. F. (2010). A neural basis for motor primitives in the spinal cord. *Journal of Neuroscience*, *30*(4), 1322–1336. http://dx.doi.org/10.1523/JNEUROSCI.5894-08.2010, URL: http://www.jneurosci.org/content/30/4/1322.

Haruno, M., Wolpert, D. M., & Kawato, M. (2003). Hierarchical MOSAIC for movement generation. In *International congress series, Vol. 1250* (pp. 575–590). Elsevier, http://dx.doi.org/10.1016/S0531-5131(03)00190-0, URL: http://www.sciencedirect.com/science/article/pii/S0531513103001900.

Hassabis, D., Kumaran, D., Summerfield, C., & Botvinick, M. (2017). Neuroscience-inspired artificial intelligence. *Neuron*, *95*(2), 245–258. http://dx.doi.org/10.1016/j.neuron.2017.06.011.

Hayakawa, T., Kamimura, T., Kaji, S., & Matsuno, F. (2020). Autonomous distributed system for gait generation for single-legged modular robots connected in various configurations. *IEEE Transactions on Robotics*, *36*(5), 1491–1510.

Heess, N., TB, D., Sriram, S., Lemmon, J., Merel, J., Wayne, G., et al. (2017). Emergence of locomotion behaviours in rich environments. arXiv preprint arXiv:1707.02286.

Heess, N., Wayne, G., Tassa, Y., Lillicrap, T., Riedmiller, M., & Silver, D. (2016). Learning and transfer of modulated locomotor controllers. arXiv preprint arXiv:1610.05182.

Heydari, S., Johnson, A., Ellers, O., McHenry, M. J., & Kanso, E. (2020). Sea star inspired crawling and bouncing. *Journal of the Royal Society Interface*, *17*(162), Article 20190700. http://dx.doi.org/10.1098/rsif.2019.0700, URL: https://royalsocietypublishing.org/doi/abs/10.1098/rsif.2019.0700. arXiv:https://royalsocietypublishing.org/doi/pdf/10.1098/rsif.2019.0700.

Huang, W., Mordatch, I., & Pathak, D. (2020). One policy to control them all: Shared modular policies for agent-agnostic control. In H. D. I.I.I., & A. Singh (Eds.), *Proceedings of machine learning research*: *vol. 119*, *Proceedings of the 37th international conference on machine learning* (pp. 4455–4464). Virtual: PMLR, URL: http://proceedings.mlr.press/v119/huang20d.html.

Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., et al. (2019). Learning agile and dynamic motor skills for legged robots. *Science Robotics*, *4*(26), eaau5872. http://dx.doi.org/10.1126/scirobotics.aau5872, URL: http://robotics.sciencemag.org/lookup/doi/10.1126/scirobotics.aau5872.

Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, *21*(4), 642–653. http://dx.doi.org/10.1016/j.neunet.2008.03.014.

Ijspeert, A. J. (2014). Biorobotics: using robots to emulate and investigate agile locomotion. *Science*, *346*(6206), 196–203. http://dx.doi.org/10.1126/science.1254486.

Ijspeert, A. J. (2018). Decoding the neural mechanisms underlying locomotion using mathematical models and bio-inspired robots: From lamprey to human locomotion. In A. Bicchi, & W. Burgard (Eds.), *Robotics research: Volume 1* (pp. 177–186). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-319-51532-8_11.

Inagaki, S., Yuasa, H., & Arai, T. (2003). CPG model for autonomous decentralized multi-legged robot system—generation and transition of oscillation patterns and dynamics of oscillators. *Robotics and Autonomous Systems*, *44*(3–4), 171–179.

Jindrich, D. L., & Full, R. J. (2002). Dynamic stabilization of rapid hexapedal locomotion. *The Journal of Experimental Biology*, *205*(Pt 18), 2803–2823.

Kalakrishnan, M., Buchli, J., Pastor, P., Mistry, M., & Schaal, S. (2010). Fast, robust quadruped locomotion over challenging terrain. In *2010 IEEE international conference on robotics and automation* (pp. 2665–2670). http://dx.doi.org/10.1109/ROBOT.2010.5509805.

Kano, T., Kanauchi, D., Ono, T., Aonuma, H., & Ishiguro, A. (2019). Flexible coordination of flexible limbs: Decentralized control scheme for inter- and intra-limb coordination in brittle stars' locomotion. *Frontiers in Neurorobotics*, *13*, 104. http://dx.doi.org/10.3389/fnbot.2019.00104, URL: https://www.frontiersin.org/article/10.3389/fnbot.2019.00104.

Kidziński, Ł., Mohanty, S. P., Ong, C. F., Huang, Z., Zhou, S., Pechenko, A., et al. (2018). Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments. In *The NIPS'17 competition: Building intelligent systems* (pp. 121–153). Springer.

Kim, J., Alspach, A., & Yamane, K. (2017). Snapbot: a reconfigurable legged robot. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 5861–5867). IEEE.

Kim, S., & Wensing, P. M. (2017). Design of dynamic legged robots. *Foundations and Trends® in Robotics*, *5*(2), 117–190. http://dx.doi.org/10.1561/2300000044.

Klimov, O., & Schulman, J. (2017). Roboschool. URL: https://openai.com/blog/roboschool/.

Konen, K., Korthals, T., Melnik, A., & Schilling, M. (2019). Biologically-inspired deep reinforcement learning of modular control for a six-legged robot. In *2019 IEEE international conference on robotics and automation workshop on learning legged locomotion workshop,(ICRA) 2019, Montreal, CA, May 20-25, 2019*.

Kruskal, W. H., & Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, *47*(260), 583–621.

Kulkarni, T., Narasimhan, K., Saeedi, A., & Tenenbaum, J. B. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems* (pp. 3675–3683).

Kuo, A. D. (2002). The relative roles of feedforward and feedback in the control of rhythmic movements. *Motor Control*, *6*(2), 129–145.

Kurach, K., Raichuk, A., Stańczyk, P., Zajkac, M., Bachem, O., Espeholt, L., et al. (2019). Google research football: A novel reinforcement learning environment. arXiv:1907.11180 [cs, stat]. URL: http://arxiv.org/abs/1907.11180.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, *40*.

Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., et al. (2017). A unified game-theoretic approach to multiagent reinforcement learning. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 4193–4206). Red Hook, NY, USA: Curran Associates Inc..

Laversanne-Finot, A., Péré, A., & Oudeyer, P. (2018). Curiosity driven exploration of learned disentangled goal spaces. CoRR, abs/1807.01521. URL: http://arxiv.org/abs/1807.01521. arXiv:1807.01521.

Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, *17*(1), 1334–1373.

Liang, E., Liaw, R., Moritz, P., Nishihara, R., Fox, R., Goldberg, K., et al. (2018). RLlib: Abstractions for distributed reinforcement learning. arXiv:1712.09381 [cs]. arXiv:1712.09381.

Liang, J., Makoviychuk, V., Handa, A., Chentanez, N., Macklin, M., & Fox, D. (2018). GPU-accelerated robotic simulation for distributed reinforcement learning. In *Conference on robot learning* (pp. 270–282).

Lin, Z., Yang, D., Zhao, L., Qin, T., Yang, G., & Liu, T.-Y. (2020). RD^2: Reward decomposition with representation decomposition. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems, Vol. 33* (pp. 11298–11308). Curran Associates, Inc., URL: https://proceedings.neurips.cc/paper/2020/file/82039d16dce0aab3913b6a7ac73deff7-Paper.pdf.

Lin, Z., Zhao, L., Yang, D., Qin, T., Yang, G., & Liu, T.-Y. (2019). Distributional reward decomposition for reinforcement learning. arXiv:1911.02166.

Lipson, H. (2019). Robots on the run. *Nature*, *568*(7751), 174. http://dx.doi.org/10.1038/d41586-019-00999-w, URL: http://www.nature.com/articles/d41586-019-00999-w.

Magill, R., & Anderson, D. I. (2017). *Motor learning and control: Concepts and applications*. McGraw-Hill Education.

McClelland, J. L. (2009). The place of modeling in cognitive science. *Topics in Cognitive Science*, *1*(1), 11–38.

McGeer, T. (1993). Dynamics and control of bipedal locomotion. *Journal of Theoretical Biology*, *163*, 277–314.

Mengistu, H., Huizinga, J., Mouret, J.-B., & Clune, J. (2016). The evolutionary origins of hierarchy. *PLOS Computational Biology*, *12*(6), Article e1004829.

Merel, J., Botvinick, M., & Wayne, G. (2019). Hierarchical motor control in mammals and machines. *Nature Communications*, *10*(1), 1–12. http://dx.doi.org/10.1038/s41467-019-13239-6, URL: https://www.nature.com/articles/s41467-019-13239-6. 00000.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. http://dx.doi.org/10.1038/nature14236.

More, H. L., & Donelan, J. M. (2018). Scaling of sensorimotor delays in terrestrial mammals. *Proceedings: Biological Sciences*, *285*(1885), http://dx.doi.org/10.1098/rspb.2018.0613.

Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., et al. (2018). Ray: A distributed framework for emerging AI applications. arXiv:1712.05889 [cs, stat]. arXiv:1712.05889.

Mountcastle, V. B. (1997). The columnar organization of the neocortex.. *Brain*, *120*(4), 701–722. http://dx.doi.org/10.1093/brain/120.4.701, arXiv:https://academic.oup.com/brain/article-pdf/120/4/701/17863573/1200701.pdf.

Mozifian, M., Higuera, J. C. G., Meger, D., & Dudek, G. (2019). Learning domain randomization distributions for training robust locomotion policies. arXiv:1906.00410 [cs, stat]. URL: http://arxiv.org/abs/1906.00410.

Najarro, E., & Risi, S. (2020). Meta-learning through hebbian plasticity in random networks. In *Advances in neural information processing systems 33: Annual conference on neural information processing systems 2020* (p. 13).

Neftci, E. O., & Averbeck, B. B. (2019). Reinforcement learning in artificial and biological systems. *Nature Machine Intelligence*, *1*, 133–143. http://dx.doi.org/10.1038/s42256-019-0025-4, URL: https://www.nature.com/articles/s42256-019-0025-4.

Neveln, I. D., Tirumalai, A., & Sponberg, S. (2019). Information-based centralization of locomotion in animals and robots. *Nature Communications*, *10*(1), 1–11. http://dx.doi.org/10.1038/s41467-019-11613-y, URL: https://www.nature.com/articles/s41467-019-11613-y.

Nishii, J. (2006). An analytical estimation of the energy cost for legged locomotion. *Journal of Theoretical Biology*, *238*(3), 636–645. http://dx.doi.org/10.1016/j.jtbi.2005.06.027, URL: http://www.sciencedirect.com/science/article/pii/S0022519305002857.

Nishikawa, K., Biewener, A. A., Aerts, P., Ahn, A. N., Chiel, H. J., Daley, M. A., et al. (2007). Neuromechanics: an integrative approach for understanding motor control. *Integrative and Comparative Biology*, *47*(1), 16–54. http://dx.doi.org/10.1093/icb/icm024.

Niven, J. E., Ott, S. R., & Rogers, S. M. (2012). Visually targeted reaching in horseshead grasshoppers. *Proceedings of the Royal Society B: Biological Sciences*, *279*(1743), 3697–3705. http://dx.doi.org/10.1098/rspb.2012.0918.

Owaki, D., & Ishiguro, A. (2017). A quadruped robot exhibiting spontaneous gait transitions from walking to trotting to galloping. In *Scientific reports*. http://dx.doi.org/10.1038/s41598-017-00348-9.

Paskarbeit, J., Schilling, M., Schmitz, J., & Schneider, A. (2015). Obstacle crossing of a real, compliant robot based on local evasion movements and averaging of stance heights using singular value decomposition. In *2015 IEEE international conference on robotics and automation (ICRA)* (pp. 3140–3145).

Pearson, K. G. (1995). Proprioceptive regulation of locomotion. *Current Opinion in Neurobiology*, *5*(6), 786–791. http://dx.doi.org/10.1016/0959-4388(95)80107-3, URL: http://www.sciencedirect.com/science/article/pii/0959438895801073.

Peng, X. B., Berseth, G., Yin, K., & Van De Panne, M. (2017). DeepLoco: dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics*, *36*(4), 1–13. http://dx.doi.org/10.1145/3072959.3073602, URL: http://dl.acm.org/citation.cfm?doid=3072959.3073602.

Peng, X. B., Coumans, E., Zhang, T., Lee, T.-W., Tan, J., & Levine, S. (2020). Learning agile robotic locomotion skills by imitating animals. arXiv:2004.00784 [cs]. arXiv:2004.00784.

Raff, E. (2019). A step toward quantifying independently reproducible machine learning research. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 5486–5496). Curran Associates, Inc., URL: http://papers.nips.cc/paper/8787-a-step-toward-quantifying-independently-reproducible-machine-learning-research.pdf.

Rao, N., Aljalbout, E., Sauer, A., & Haddadin, S. (2020). How to make deep RL work in practice. arXiv:2010.13083 [cs]. arXiv:2010.13083.

Reda, D., Tao, T., & van de Panne, M. (2020). Learning to locomote: Understanding how environment design matters for deep reinforcement learning. In *Proc. ACM SIGGRAPH conference on motion, interaction and games*.

Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M., Hadsell, R., et al. (2018). Graph networks as learnable physics engines for inference and control. arXiv:1806.01242 [cs, stat]. URL: http://arxiv.org/abs/1806.01242.

Schilling, M., & Cruse, H. (2007). Hierarchical MMC networks as a manipulable body model. In *Inproceedings* (pp. 2141–2146).

Schilling, M., & Cruse, H. (2017). ReaCog, a minimal cognitive controller based on recruitment of reactive systems. *Frontiers in Neurorobotics*, *11*, http://dx.doi.org/10.3389/fnbot.2017.00003.

Schilling, M., & Cruse, H. (2020). Decentralized control of insect walking: A simple neural network explains a wide range of behavioral and neurophysiological results. *PLoS Computational Biology*, *16*(4), Article e1007804. http://dx.doi.org/10.1371/journal.pcbi.1007804.

Schilling, M., Cruse, H., & Arena, P. (2007). Hexapod Walking: an expansion to Walknet dealing with leg amputations and force oscillations. *Biological Cybernetics*, *96*(3), 323–340. http://dx.doi.org/10.1007/s00422-006-0117-1.

Schilling, M., Hoinville, T., Schmitz, J., & Cruse, H. (2013). Walknet, a bio-inspired controller for hexapod walking. *Biological Cybernetics*, *107*(4), 397–419. http://dx.doi.org/10.1007/s00422-013-0563-5.

Schilling, M., Konen, K., Ohl, F. W., & Korthals, T. (2020). Decentralized deep reinforcement learning for a distributed and adaptive locomotion controller of a hexapod robot. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*. Las Vegas, NV, USA (Virtual) (p. 8).

Schilling, M., & Melnik, A. (2018). An approach to hierarchical deep reinforcement learning for a decentralized walking control architecture. In *Biologically inspired cognitive architectures 2018. proceedings of the ninth annual meeting of the BICA society, 848*. URL: https://pub.uni-bielefeld.de/record/2934190.

Schilling, M., Paskarbeit, J., Hoinville, T., Hüffmeier, A., Schneider, A., Schmitz, J., et al. (2013). A hexapod walker using a heterarchical architecture for action selection. *Frontiers in Computational Neuroscience*, *7*, 126. http://dx.doi.org/10.3389/fncom.2013.00126.

Schilling, M., Paskarbeit, J., Ritter, H., Schneider, A., & Cruse, H. (2021). From adaptive locomotion to predictive action selection – Cognitive control for a six-legged walker. *IEEE Transactions on Robotics*, 1–17. http://dx.doi.org/10.1109/TRO.2021.3106832.

Schilling, M., Paskarbeit, J., Schmitz, J., Schneider, A., & Cruse, H. (2012). Grounding an internal body model of a hexapod walker — control of curve walking in a biological inspired robot. In *2012 IEEE/RSJ international conference on intelligent robots and systems* (pp. 2762–2768).

Schilling, M., Ritter, H., & Ohl, F. W. (2019). From crystallized adaptivity to fluid adaptivity in deep reinforcement learning — Insights from biological systems on adaptive flexibility. In *IEEE international conference on systems, man, and cybernetics 2019*.

Schmitz, J., Schneider, A., Schilling, M., & Cruse, H. (2008). No need for a body model: Positive velocity feedback for the control of an 18-DOF robot walker. *Applied Bionics and Biomechanics, Special Issue on Biologically Inspired Robots*, *5*(3), 135–147.

Schneider, J., Wong, W.-K., Moore, A., & Riedmiller, M. (1999). Distributed value functions. In *Proceedings of the sixteenth international conference on machine learning* (pp. 371–378). Morgan Kaufmann.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2015). Trust region policy optimization. CoRR, abs/1502.05477. URL: http://arxiv.org/abs/1502.05477. arXiv:1502.05477.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Seijen, H. V., Fatemi, M., Laroche, R., Romoff, J., Barnes, T., & Tsang, J. (2017). Hybrid reward architecture for reinforcement learning. In *NIPS*.

Semini, C., Tsagarakis, N. G., Guglielmino, E., Focchi, M., Cannella, F., & Caldwell, D. G. (2011). Design of HyQ – a hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, *225*(6), 831–849. http://dx.doi.org/10.1177/0959651811402275, arXiv:https://doi.org/10.1177/0959651811402275.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv:1312.6034 [cs]. URL: http://arxiv.org/abs/1312.6034.

Sponberg, S., & Full, R. J. (2008). Neuromechanical response of musculo-skeletal structures in cockroaches during rapid running on rough terrain. *The Journal of Experimental Biology*, *211*(Pt 3), 433–446. http://dx.doi.org/10.1242/jeb.012385.

Sprague, N., & Ballard, D. H. (2003). Multiple-goal reinforcement learning with modular sarsa(0). In G. Gottlob, & T. Walsh (Eds.), *IJCAI-03, Proceedings of the eighteenth international joint conference on artificial intelligence, Acapulco, Mexico, August 9-15, 2003* (pp. 1445–1447). Morgan Kaufmann, URL: http://ijcai.org/Proceedings/03/Papers/233.pdf.

Steingrube, S., Timme, M., Wörgötter, F., & Manoonpong, P. (2010). Self-organized adaptation of a simple neural circuit enables complex robot behaviour. *Nature Physics*, Advanced Online Publication, January 17.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). The MIT Press, URL: http://incompleteideas.net/book/the-book-2nd.html.

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., et al. (2018). Sim-to-real: learning agile locomotion for quadruped robots. arXiv:1804.10332 [cs]. arXiv:1804.10332.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., et al. (2018). Deepmind control suite. CoRR, abs/1801.00690. URL: http://arxiv.org/abs/1801.00690. arXiv:1801.00690.

Theunissen, L. M., Vikram, S., & Dürr, V. (2014). Spatial co-ordination of foot contacts in unrestrained climbing insects. *Journal of Fish Biology*, *217*(18), 3242–3253. http://dx.doi.org/10.1242/jeb.108167, URL: https://jeb.biologists.org/content/217/18/3242.

Todorov, E., Erez, T., & Tassa, Y. (2012). MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems* (pp. 5026–5033).

Tsounis, V., Alge, M., Lee, J., Farshidian, F., & Hutter, M. (2020). DeepGait: planning and control of quadrupedal gaits using deep reinforcement learning. *IEEE Robotics and Automation Letters*, *5*(2), 3699–3706. http://dx.doi.org/10.1109/LRA.2020.2979660.

Uithol, S., van Rooij, I., Bekkering, H., & Haselager, P. (2012). Hierarchies in action and motor control. *Journal of Cognitive Neuroscience*, *24*(5), 1077–1086. http://dx.doi.org/10.1162/jocn_a_00204.

Wang, T., Liao, R., Ba, J., & Fidler, S. (2018). Nervenet: Learning structured policy with graph neural networks. In *International conference on learning representations*.

Webb, B. (2001). Can robots make good models of biological behaviour? *Behavioral and Brain Sciences*, *24*(6).

Whitman, J., Su, S., Coros, S., Ansari, A., & Choset, H. (2017). Generating gaits for simultaneous locomotion and manipulation. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 2723–2729). IEEE.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818–833). Springer.

Zhang, C., Vinyals, O., Munos, R., & Bengio, S. (2018). A study on overfitting in deep reinforcement learning. arXiv:1804.06893 [cs, stat]. arXiv:1804.06893.

Zill, S. N., & Moran, D. T. (1981). The exoskeleton and insect proprioception: III. Activity of tribal campaniform sensilla during walking in the American cockroach, periplaneta Americana. *Journal of Fish Biology*, *94*(1), 57–75, URL: https://jeb.biologists.org/content/94/1/57.

Zill, S., Schmitz, J., & Büschges, A. (2004). Load sensing and control of posture and locomotion. *Arthropod Structure & Development*, *33*, 273–286.