Mechatronics 34 (2016) 1–11

Contents lists available at ScienceDirect

Mechatronics

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

Intelligent laser welding through representation, prediction, and control learning: An architecture with deep neural networks and reinforcement learning

Johannes Günther [a,*], Patrick M. Pilarski [b], Gerhard Helfrich [a], Hao Shen [a], Klaus Diepold [a]

[a] Dept. Electrical and Computer Engineering, Technische Universität München, Munich 80290, Germany
[b] Dept. of Medicine, University of Alberta, Edmonton, AB T6G 2E1, Canada

ARTICLE INFO

Article history:
Received 15 October 2014
Revised 23 July 2015
Accepted 4 September 2015
Available online 1 October 2015

Keywords:
Deep learning
Reinforcement learning
Prediction
Control
Laser welding

ABSTRACT

Laser welding is a widely used but complex industrial process. In this work, we propose the use of an integrated machine intelligence architecture to help address the significant control difficulties that prevent laser welding from seeing its full potential in process engineering and production. This architecture combines three contemporary machine learning techniques to allow a laser welding controller to learn and improve in a self-directed manner. As a first contribution of this work, we show how a deep, auto-encoding neural network is capable of extracting salient, low-dimensional features from real high-dimensional laser welding data. As a second contribution and novel integration step, these features are then used as input to a temporal-difference learning algorithm (in this case a general-value-function learner) to acquire important real-time information about the process of laser welding; temporally extended predictions are used in combination with deep learning to directly map sensor data to the final quality of a welding seam. As a third contribution and final part of our proposed architecture, we suggest that deep learning features and general-value-function predictions can be beneficially combined with actor–critic reinforcement learning to learn context-appropriate control policies to govern welding power in real time. Preliminary control results are demonstrated using multiple runs with a laser-welding simulator. The proposed intelligent laser-welding architecture combines representation, prediction, and control learning: three of the main hallmarks of an intelligent system. As such, we suggest that an integration approach like the one described in this work has the capacity to improve laser welding performance without ongoing and time-intensive human assistance. Our architecture therefore promises to address several key requirements of modern industry. To our knowledge, this architecture is the first demonstrated combination of deep learning and general value functions. It also represents the first use of deep learning for laser welding specifically and production engineering in general. We believe that it would be straightforward to adapt our architecture for use in other industrial and production engineering settings.

1. Introduction

Laser welding is a precise and fast welding technique that sees widespread use in industrial welding systems [1]. Unfortunately, laser welding is a complex process that is often hard to control [2]. To address control difficulties, recent research has demonstrated cognitive laser welding systems that perform well on a defined work piece after setup [3]. Nevertheless, cognitive control is still in an early stage of development [4], and to fulfill the requirements of modern industry, systems must have the flexibility to deal with changing conditions without the need for demanding and time-intensive manual setup [5].

To address the need for both rapid setup times and welding system flexibility, we propose the idea of a self-learning and self-improving laser-welding system that would be able to perform well under changing circumstances. As a classical model-based approach is not feasible due to the dynamics and uncertainty inherent to the process, we suggest applying machine learning techniques. Our proposed approach brings together a selection of modern machine learning techniques, including deep-learning neural networks for generating state representations and state-of-the-art reinforcement learning prediction and control algorithms. These algorithms empower the system to leverage

* Corresponding author. Tel.: +49 89 289 23631; fax: +49 289 23600.
E-mail address: johannes.guenther@tum.de (J. Günther).

http://dx.doi.org/10.1016/j.mechatronics.2015.09.004
0957-4158/

important aspects of intelligence during welding, namely perception, prediction, and interaction.

Representation: As a laser-welding system's sensor signals are multidimensional and multimodal, it is often not realistic to use them directly as an input for real-time control learning algorithms. Building on established ideas in dimensionality reduction, we therefore use a representation-learning (perception) algorithm to transform the raw sensor data into a low-dimensional and transformation-invariant representation of the systems state. The system learns to abstract its inputs. In particular, a technique that has shown its capability to produce the lowest classification error for various problems when used for feature extraction is deep learning [6]. Furthermore, deep auto-encoders have been shown to successfully compete with state-of-the art feature extraction techniques (e.g., principal component analysis, linear discriminant analysis) [7] and improved [8] or directly learned [9,10] policies for high-dimensional image data in reinforcement learning. Stacked denoising auto-encoders have shown the capability of achieving a general representation, which leads to more robustness against varying data and overfitting [11].

Prediction: A very common problem in industry is the inability to directly measure process quality. There are several approaches to this issue, e.g., system models, envelope curves or look-up tables. But these techniques are restricted either in applicability (a priori model), accuracy (envelope curves) or scalability (look-up tables). They are also limited in their capability to adapt to changes. To deal with these issues, we include predictions about process quality and state as an important part of intelligence [12] in our architecture; importantly, we suggest that predictions should be able to be learned and adapted during the ongoing operation of a system. To date, prediction learning has been dominated by linear models that are difficult to apply to nonlinear and time-varying problems [13]. These problems have been overcome by recent research using the temporal-difference (TD) reinforcement learning approach [14]. New techniques have extended classical TD-learning to allow generalized online predictions [15]. We include these predictions into our proposed system using a temporally extended prediction approach called nexting [16] with general value functions, an approach that is capable of learning and making real-time predictions at multiple timescales.

Control: There exist a number of different controllers for industrial applications, e.g., PID-controllers, adaptive controllers and fuzzy controllers. Given a correct and accessible quality measurement, it would be easy to implement these techniques for laser welding. But all these approaches need a time-consuming and human assisted setup process and do not work well for changing conditions. To enable our architecture to provide a high-quality welding seam on its own, it is necessary to have a controller that can learn from experience and improve its own performance. Therefore we suggest a machine learning algorithm, namely an actor–critic reinforcement learning (ACRL) algorithm [17]. This type of algorithm consists of two parts: an actor and a critic. The actor takes actions according to a learned policy while the critic evaluates these actions. The actor–critic algorithm has several characteristics that are useful for our specific control problem. As ACRL algorithms are parameter based, their computation can be done incrementally (linearly) and they can be updated within milliseconds. Due to the fact that experience—from which the algorithm already had learned—does not need to be stored, the memory requirements do not increase over time [18]. By using function approximation they also scale well to real world problems; this has been shown in various applications [19–22].

Our proposed architecture [23] for integrating representation, prediction and control in laser welding therefore promises to address key industry needs relating to both the calibration and optimization of diverse welding processes. It is described in the remainder of this manuscript as follows. Section 2 describes the laser welding system and the monitoring, as well as how the algorithms will work together in the proposed architecture. Section 3 focuses on deep learning and how features are generated via deep auto-encoders from the existing sensor input. These features are the input for the reinforcement learning algorithms, explained and evaluated in Section 4. The results are discussed in Section 5 and followed by concluding remarks in Section 6.

2. Laser welding and the proposed architecture

2.1. The laser welding process and monitoring

Although laser welding is quite common in industrial applications, it is still necessary to closely and consistently monitor and control the process [24]. Despite the environmental uncertainties that the process is exposed to, like changes in temperature, humidity or the welding gas quality, there are also uncertainties caused by the material. These include, but are not limited to, changes in the chemical compounding, and the thickness and contamination of the surface. Fig. 1 illustrates examples for laser welds with different quality.

In our setting, process monitoring is done by a camera-based system and photodiodes, which is a common setting in laser welding applications [25]. As the keyhole, which is the area where the laser hits the material, oscillates with a typical frequency of 500 Hz [26], all sensors have to sample with at least twice this frequency. This can be considered as a benchmark real-time capability for the process. The camera can sample at rates of up to 1500 Hz. It provides important information about geometrical parameters of the observed keyhole [27] with a resolution of 144×176 pixels. Additionally, the process is observed by three photodiodes, sampling at 40 kHz and corresponding to different wavelengths. The first diode observes the process temperature at the wavelength between 1100 nm and 1800 nm. The second observes the plasma radiation at a wavelength of 400–600 nm. The third diode records the laser back reflection at 1050–1080 nm.

2.2. Architecture

Laser welding is a dynamic process with high uncertainty and therefore it is not feasible to build a precise model of the process, which would be the classical control approach. We therefore propose a machine learning approach. Our suggested architecture combines deep neural networks (DNN) [7] with reinforcement learning algorithms [28].

Fig. 2 shows the architecture, which consists of three parts: representation, process knowledge (prediction), and process control. In the first part—deep learning of representations—the monitored sensor data is processed and transformed into informative features which are lower in dimension to ensure real-time capability and robustness. By doing so, the system is able to detect its current state only by the provided sensor data and is therefore more invariant to environmental changes. These features are used in the second part—prediction—to build up knowledge about the process. By using temporally extended predictions, the system has the capacity to evaluate its current performance and predict how its actions might impact its performance in the future. The features from the representation and the knowledge from the second part are combined in the third part—process control—to control the system in terms of the laser power applied to the welding surface.

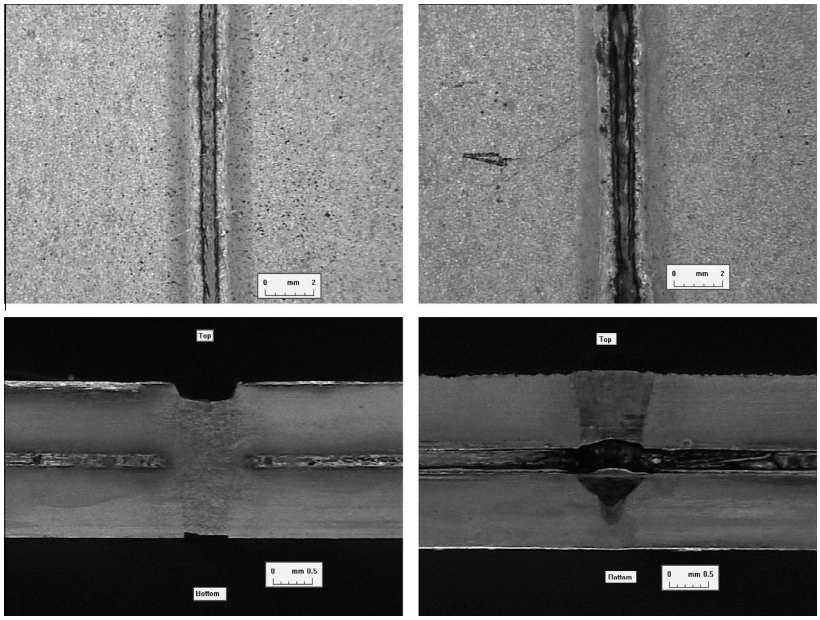

Fig. 1. Laser welds and corresponding cross sections for zinc-coated steel in overlap position. On the left side is a laser weld of high quality, while the weld on the right side is a failure. In the upper row the difference can be distinguished by the smoothness of the welding seam. In the cross section it is clearly visible that the laser weld on the right sight is not sufficient. There is no connection between the two plates.

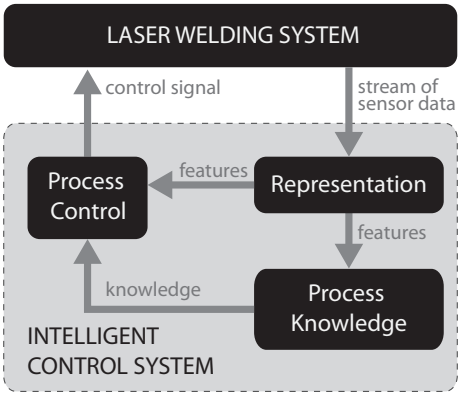

Fig. 2. Proposed architecture for intelligent laser welding. The architecture consists of three elements: representation, process knowledge (prediction), and process control. The first element extracts meaningful low-dimensional features from the sensor data. The second element then uses these features to learn about the process and build up knowledge. This knowledge together with the features is used to control the process in the third element.

3. Deep learning

Our representation approach employs techniques from deep learning. Deep learning is a concept inspired by the way the visual cortex of mammals is structured [29]. The visual cortex is built in a hierarchical way—in the first representation layer, simple edges are detected and then gradually combined to more abstract features in higher layers, leading to a robust representation. To benefit from this concept, artificial neural networks with several layers of non-linearity were created. Adding more layers, however, leads to more complex and non-convex optimization problems, which can cause this approach to perform worse than shallow neural networks [30]. This problem has been overcome by a greedy layer-wise pre-training, where first each layer is trained individually (pre-training), followed by fine tuning performed on the whole neural network [7].

Given the right architecture, i.e., reducing the number of neurons in subsequent layers, this learning technique can not only be used to generate meaningful representations for a given data set, but also to compress it. In order to create compressed features, the neural network has to reduce the number of neurons in the middle layer. This architecture is called an auto-encoder [7]. In its simplest form, an auto-encoder consists of three layers, namely, the input layer, the hidden representation layer and the output layer. We assume that there are p neurons in the input and output layers, and q neurons in the hidden layer. Let us denote by $f : \mathbb{R} \to \mathbb{R}$ the activation function, e.g., the sigmoid function, and $x \in \mathbb{R}^p$ a vector from the input layer. The feature value at the i-th neuron in the hidden layer is then computed by

$$h_i(x) := f\left(w_i^\top x + b_i\right), \qquad \text{for all } i = 1, \ldots, q, \tag{1}$$

where $w_i \in \mathbb{R}^p$ denotes the weighting coefficients associated with the i-th neuron, and $b_i \in \mathbb{R}$ is the corresponding bias (offset). After calculating the features in the hidden layer, the hidden representation will serve as an input for the decoding/output layer. Let us denote by

$$h := \left[h_1(x), h_2(x), \ldots, h_q(x)\right]^\top \in \mathbb{R}^q \tag{2}$$

the representation vector in the hidden layer. Then, the computation in the output layer is done as follows

$$y_j(h) := f'\left(w_j'^\top h + b_j'\right), \qquad \text{for all } j = 1, \ldots, p, \tag{3}$$

where $w_j' \in \mathbb{R}^q$ and $b_j' \in \mathbb{R}$ are the parameters associated to the j-th neuron in the output layer. Let us denote $W := [w_1, \ldots, w_q] \in \mathbb{R}^{p \times q}$ and $W' := [w_1', \ldots, w_p] \in \mathbb{R}^{q \times p}$. Note that the activation function f' is not required to be the same activation function as f. If the decoder weights W' are tied to the encoder weights W, i.e., $W' = W^\top$, the working of an auto-encoder is comparable to the behavior of a Restricted Boltzmann Machine [11].

The error between the original input vector x and the reconstruction $y := \left[y_1(h), \ldots, y_p(h)\right]^T \in \mathbb{R}^p$ can be considered as a measure for the quality of the hidden representation h and serves as loss function for the backpropagation algorithm [31]. For real-valued inputs, e.g., images, the mean squared error

(MSE) is the most common choice [11], resulting in the final loss function

$$L(x, y, W, W') = \frac{1}{2}\|x - y\|_2^2 + \frac{\lambda_1}{2}\|W\|_F^2 + \frac{\lambda_2}{2}\|W'\|_F^2, \quad (4)$$

where $\|\cdot\|_F$ is the Frobenius norm of matrices. Here, the loss function consists of two parts: the residual/reconstruction error term and a regularization term, also known as a weight decay term. The weight decay parameters $\lambda_i > 0$ control the magnitude of weights in different layers in order to prevent the auto-encoder from overfitting [32].

To extend the traditional autoencoding principle to deep learning, auto-encoders are stacked on top of each other, as illustrated in Fig. 3. In this setting, the hidden representation of the first encoder serves as input for the next one, and so on. This creates representations from representations, analogous to deep belief neural networks. Following the idea of greedy layer-wise pre-training, first a traditional one-step auto-encoder is trained. After convergence, the decoding weights are fixed and another one-step auto-encoder is stacked as depicted in Fig. 3. This procedure is repeated until the full architecture is built up; this process is then followed by fine tuning over the whole stacked auto-encoder.

During layer-wise greedy pre-training, each auto-encoder input is corrupted with white Gaussian noise, sampled from a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with mean μ and standard deviation σ. White Gaussian noise, with $\mu = 0$, was chosen as it corresponds to noisy sensor readings, which occur in cameras and photodiodes. It has been shown that trying to reconstruct the uncorrupted image from corrupted inputs leads to more meaningful representations [33]. As can be seen in Table 1, the noise has a mean of zero and for the first auto-encoder a standard deviation of $\sigma = 0.5$. In the subsequent layers the standard deviation is adapted to activation of the layer to avoid unreasonably high activations due to the corruption. This is done by computing the standard deviation of the activation in each layer σ^k, where k indicates the layer, and reducing it to 10% of the activation for the noise sampling. In each layer 50% of the input neurons are corrupted with noise. An extensive introduction to stacked denoising auto-encoders can be found in [11].

Table 1
Deep auto-encoder parameters.

Parameter	Value
Neurons per layer (encoder and bottleneck)	1024–2048–1024–512–256–16
Decay learning rate	0.0005–0.00025
Momentum rate	0.005–1
L1 weight decay factor	0.00005
Mini-batch size	Pre-training: 100, fine-tuning: 200
Additive Gaussian noise in layer 1	$\mathcal{N}(0, 0.5)$
Additive Gaussian noise in layer 2–4	$\mathcal{N}(0, 0.1\sigma^k)$
Corruption rate	50%

As activation function for the encoder we selected a partially linear function that has more desirable properties than the typical logistic or tanh activation functions [34]. A unit that uses this activation function is called a *rectifier linear unit* (ReLU). The ReLU activation function is defined as

$$f(x) = \max(x, 0), \quad (5)$$

and as a consequence, roughly 50% of the units in the hidden layers are off for random initial weights. Therefore ReLUs learn a sparser representation compared to other units. Further, ReLUs allow faster training in a twofold sense: they need less iterations to progress and the execution of their activation function is much faster on most CPUs [35]. The decoder uses a non-squashing activation function, as suggested for real-valued inputs [11]. The choice of the learning rate has a crucial influence on the convergence and accuracy of results. The learning rate decreases and the momentum increases linearly, as described in [36,37], after half of the iterations for both pre-training and fine tuning.

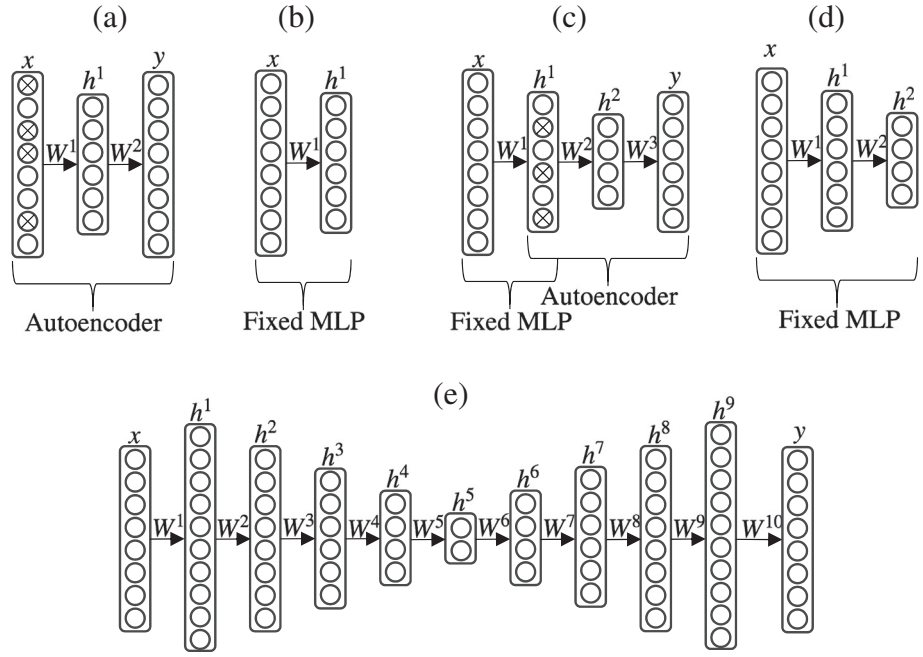

Fig. 3. Stacked denoising auto-encoder. (a) An auto-encoder with only one hidden layer learns the mapping from a corrupted input to the output. (b) As soon as learning converges the weights are fixed. (c) A new, smaller, layer is stacked on top of the first layer. The inputs to the second hidden layer are corrupted with noise again. (d) After convergence, the learned weights for the first and the second hidden layers are fixed. (e) The final architecture, as used for the experiments in this work. The actual number of neurons used in each layer is described in Table 1.

To further speed up the training, we apply Mini-Batch Stochastic Gradient Descent (MB-SGD) [38] to the pre-training and fine tuning. MB-SGD is a trade-off between the iterations until convergence and the time to calculate an iteration and therefore is the fastest when the mini-batch size is optimal. For a faster convergence and lower error we apply the classical momentum method [39] to MG-SGD. Once the training process has converged, the decoding layers, i.e., all layers after the bottleneck, i.e., the auto-encoder with the smallest number of neurons for the hidden representation,which is denoted h^5 in Fig. 3e, can be removed and the hidden representation in the bottleneck layer can be used as low dimensional representations, or features, as depicted in Fig. 2.

To test the performance of the DNN approach described above, experiments were conducted on 16,000 laser welding images from different processes, divided into four sets of equal size. These sets were used for fourfold cross-validation. A region of interest was applied to each image and the section was subsampled to a size of 32×32. For the hidden representation in the bottleneck experiments with $q \in \{4, 8, 16, 32, 64\}$ were done. Using 16 features yielded the lowest error. The deep auto-encoder achieved a mean value of reconstruction error of 16.6% from 16-dimensional features on 8 different test data sets. Compared to Principal Component Analysis (PCA), which had a mean reconstruction error of 15.5%, the deep auto-encoder is a competitive alternative. The deep auto-encoder features also yielded a lower classification error when used as input for two Support Vector Machine (SVM) classifiers to distinguish between clean and contaminated welding seams (see Table 2).

The purpose of the representation approach is to extract meaningful and compressed features in order to be able to distinguish different states, e.g., a clean welding seam or corrupted seam. Fig. 4 demonstrates the reaction of the neural network to different images. It can be seen that the activations vary and the neural network is therefore able to distinguish the different states presented by the images. The activation range in the bottleneck layer, as shown in Fig. 4, was not bounded in any way and is only dependent on the layer number. Due to the used activation function and the resulting sparse representation, the activation amplitude will become less with each added layer.

To give further insights into the learned DNN, we visualize its features in a specific way: all, except one, features are set to zero. The resulting output of the decoder is scaled by the value of the remaining active feature. Using this technique, we generate an image which shows the specialization and activity of the particular feature as bright parts. This is repeated for every feature. The output visualization for this process can be found in Fig. 5.

Table 2
Comparison of the SVM classification error, using PCA and deep auto-encoder extracted features with different numbers of features. The best classification results for both approaches are marked in bold.

Classification error	PCA	Deep auto-encoder
SVM RBF kernel (4 features)	4.93% ± 0.27%	5.54% ± 0.51%
SVM RBF kernel (8 features)	**1.66% ± 0.17%**	1.39% ± 0.16%
SVM RBF kernel (16 features)	1.79% ± 0.19%	**0.81% ± 0.11%**
SVM RBF kernel (32 features)	5.68% ± 0.23%	2.07% ± 0.10%
SVM RBF kernel (64 features)	No convergence	5.43% ± 0.33%

4. Reinforcement learning

Reinforcement learning (RL) is a branch of machine learning and artificial intelligence that focuses on goal-directed learning and decision making [28]. In an RL problem, the goal or objective of learning is specified in terms of a single scalar signal known as *reward* that provides a measure of success or failure. As such, methods for solving RL problems do not explicitly rely on labeled training examples during learning as in the related field of supervised learning—a RL solution method learns through trial and error. During ongoing interactions with a problem or *environment*, a reinforcement learner takes actions and observes the resulting reward. These observations are then used by the learner to change the way it selects actions. In other words, an RL solution method uses valuations of situations and situation-action pairings to alter the way it maps situations to available actions. This mapping is termed a *policy*. Because policy change is driven by the maximization of an outcome, as opposed to fitting to a desired means of achieving an outcome, RL does not necessarily require a designer to provide comprehensive domain information for each new learning scenario. RL can consider the whole problem without explicitly dividing and optimizing sub-problems. Because of this generality, and also because RL is primarily concerned with learning from ongoing sequences of experience, it is well suited for changing environments and varying conditions [28]. Integral to an RL approach are methods for learning expectations of future observations from samples of experience (prediction learning), and using samples of experience to affect policy change (control learning).

In general, a RL problem can be well thought of as a Markov Decision Process (MDP), wherein action choices by a learner partially determine the learner's progression through a set of situations known as *states*, and rewards resulting from the transition from one state to another by way of a given action. Formally, an MDP can be defined by a quadruple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, wherein $\mathcal{S}$ denotes

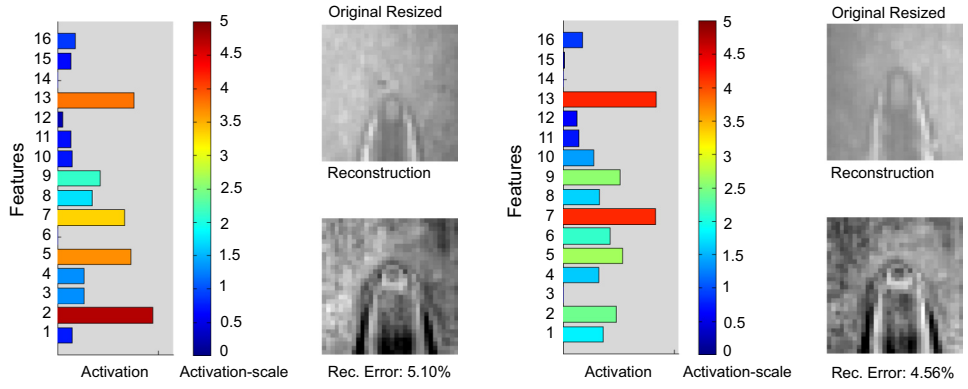

Fig. 4. Neural network reaction to different inputs. The figure shows the original image (after preprocessing), the reconstruction, and the color-coded activations for two different processes. The error is the MSE between the original image and the reconstruction. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

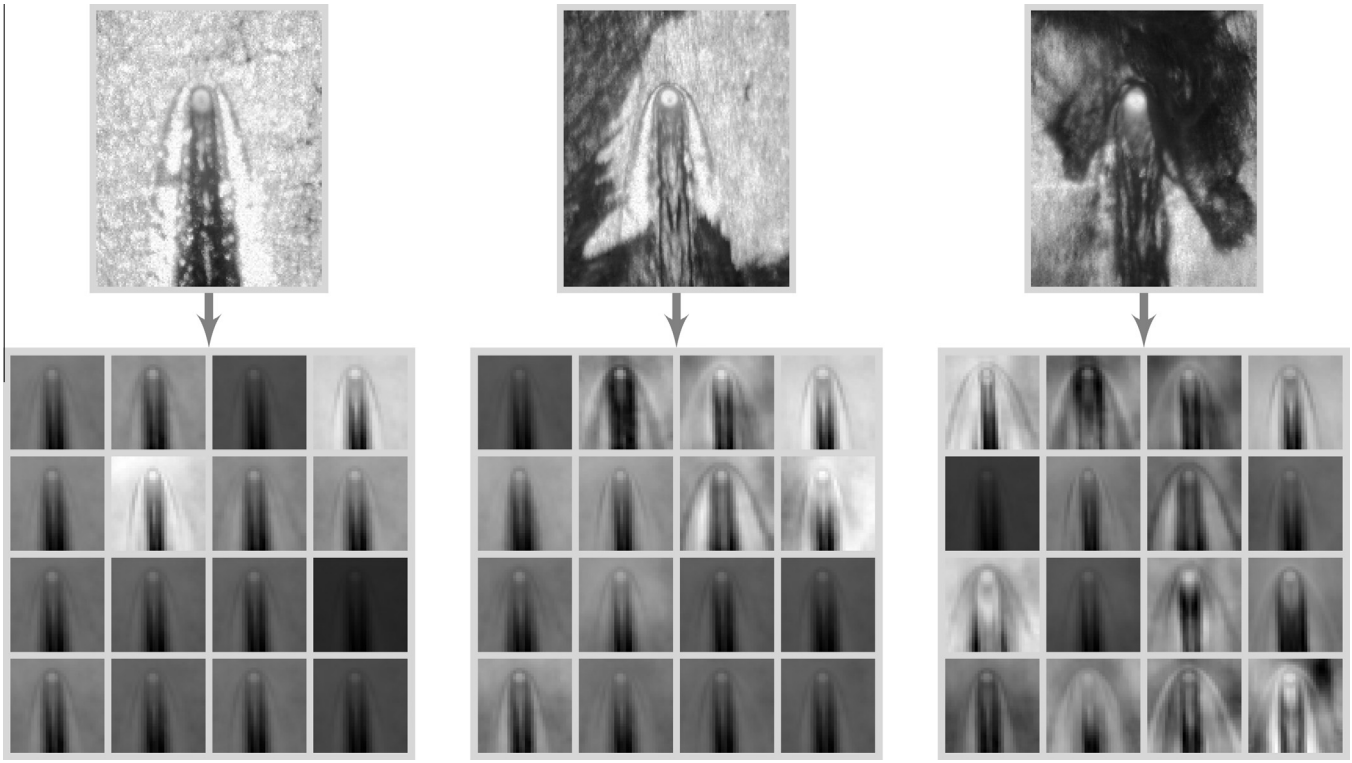

Fig. 5. DNN feature visualization and original images. The dark spots in the original image are due to contamination with oil. To improve the visibility, a luminance filter has been applied to the images. For each input image 16 features-images are reconstructed, which show the specialization and activation for the image. It can be seen that not only the level of activation but also the specialization for each feature is dependent on the input image. Different features specialize on different aspects of the input image.

the set of states, $\mathcal{A}$ the set of admissible actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ the transition probabilities between states, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ the rewards observed on these transitions. At each time step $t \in \{0, \ldots, T\}$, the learner choses an action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$, following a policy $\pi : \mathcal{S} \to \mathcal{A}$. The system will then observe a successor state $s' \in \mathcal{S}$ and reward $r = \mathcal{R}(s, a, s') \in \mathbb{R}$, according to the transition probabilities and rewards given by $\mathcal{P}$ and $\mathcal{R}$. Given an MDP, a RL system aims to maximize the expected sum of future rewards. This quantity is known as the *return*, and is typically weighted according to a discount factor $\gamma \in [0, 1]$ that determines if the learning objective places more weight on immediate (γ approaches 0) or future rewards (γ approaches 1). In order to maximize return the learner uses each sampled transition to improve a *value function*, which is a learned approximation of the true expected return for all given states ($V_\pi : \mathcal{S} \to \mathbb{R}$) or state-action pairs ($Q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$) while following policy π. The learned value function is subsequently used to improve π. This iterative process is known as *generalized policy iteration*, and is a hallmark of RL. As such, π (a control policy) and V_π (predictions of expected future reward for all states) represent the outputs or solutions learned by an RL method.

In order to represent the continuous state spaces to an RL algorithm, as in many real-world problems of interest, observations from the environment need to be approximated. One approximation technique well suited to online implementation is tile coding, which transforms continuous variables into a sparse, binary representation [28]. We employed the standard tile coding approach described by Sutton and Barto [28], giving in a function $x : \mathcal{S} \to \mathbb{B}^n$ that maps a g-dimensional column vector $s \in \mathcal{S} \subseteq \mathbb{R}^g$ to an n-dimensional binary column vector $x(s)$. V_π may then be approximated by learning a columnar weight vector w, also of dimension n; the inner product of $w^\top x(s)$ gives the learned return for any given state.

4.1. Prediction

To control a welding process reliably, it is necessary to have sufficient information about the current performance of the system. As such a performance measure is not directly available in laser welding processes, we now present our implementation of a specific RL algorithmic approach known as *nexting* [16] to estimate the effectiveness of process control. Nexting is a term known from psychology—it refers to a short-term prediction about what will happen next [40]. As demonstrated by Modayil et al., nexting also has application in computational prediction problems, and specifically presents a way to leverage well known techniques from RL to estimate the outcomes of various real-time signals [16]. In particular, nexting makes use of temporal-difference learning [28] with linear function approximation and general value functions [14], and is therefore capable of making predictions about arbitrary non-reward or reward signals at multiple time scales. Our temporal-difference learning algorithm for process quality nexting is presented in Table 3, following the approach described by Pilarski, Dick, and Sutton [22]. For this work, we evaluated two discount rates, $\gamma = 0$ and $\gamma = 0.8$. With $\gamma = 0$ the algorithm predicts the immediate pseudo-reward, i.e., the current quality, in a way analogous to incremental supervised learning using least-mean-squared updates; the choice of $\gamma = 0.8$ results in a more farsighted prediction. The trace decay parameter was set to $\lambda = 0.9$, and we used a step size (learning rate) of $\alpha = \frac{1}{m}$, where m is the number of active tiles in the tile-coded feature representation $x(s)$. Weight and trace vectors were $n = 10,001$-dimensional column vectors, and were initialized to zero.

To provide input to the learner, different laser welding processes were performed and the resulting sensor data was recorded. Later, these welds were evaluated by an expert and labeled, which means each process is classified by its quality, according to EN ISO

Table 3
Nexting algorithm for use in laser welding process quality prediction.

Algorithm 1		Nexting with Temporal-Difference Learning	
1: **initialize:** w, e, s			
2: **repeat:**			
3: **observe** r, s'			
4: $\delta \leftarrow r + \gamma w^\top x(s') - w^\top x(s)$		// calculate the td-error, based on the current weights	
5: $e \leftarrow \gamma\lambda e + x(s)$		// update the eligibility trace e, based on decay and visited state	
6: $w \leftarrow w + \alpha\delta e$		// update the weight vector w based on the td-error	
7: $s \leftarrow s'$		// make successor state the current state	
$w \in \mathbb{R}^n$:	Value function weight vector	$e \in \mathbb{R}^n$:	Eligibility trace vector
$s \in \mathcal{S} \subseteq \mathbb{R}^k$:	Current state vector	$s' \in \mathcal{S} \subseteq \mathbb{R}^k$:	Successor state vector
$r \in \mathbb{R}$:	Reward	$\gamma \in [0, 1]$:	Discount factor
$x(s) \in \mathbb{B}^n$:	Tile coded state vector	$\alpha \in (0, 2)$:	Learning rate
$\lambda \in [0, 1]$:	Eligibility trace decay factor		

13919-1:1996. The labels range from 1 to 4, where 1 indicates a non-sufficient laser weld and 4 denotes the ideal laser weld, denoted as B in the EN ISO 13919-1:1996. The labels 2 and 3 correspond to the classes D and C, respectively. These expert-derived quality measures served as the signal to be predicted by the nexting algorithm—i.e., the pseudo-reward signal r used as part of the temporal-difference update in line 4 of Table 3. Based on the current state—i.e., the sensor data, denoted s—the algorithm calculates a predicted value, using its weight vector w. This value, $w^\top x(s)$, is then compared with the actual quality r and the discounted prediction for the next observed state $\gamma w^\top x(s')$ to compute the scalar temporal-difference error δ (td-error δ, line 4). The weights are then shifted so as to reduce the td-error (line 6), improving the algorithm's prediction such that the algorithm approaches the td-fixpoint: $r + \gamma w^\top x(s') = w^\top x(s)$. This update is done sequentially for each time step in the dataset. As a result of this training process, the weight vector w becomes a compact summary of the feature-to-quality relationships underlying the labeled examples. After the weight vector is learned appropriately, the learned weights, and thus the stored predictions, can be applied to new processes. For visual comparison with the labeled process quality values, the learned return $w^\top x(s)$ may be γ-normalized according to the degree of discounting using $(1-\gamma)w^\top x(s)$.

The algorithm was trained on a zinc-coated laser welding process in an overlap position. The laser was set to the appropriate conditions for the material, i.e., a power of 2000 W and a velocity of 3.5 m/min. The welding seam was contaminated with grease on two different spots. The contamination impacts the quality of welding, as the grease starts to burn once the laser hits it. This most likely results in an insufficient joint and therefore in a drop of the welding quality for this area. For each experiment we evaluated our approach using two error measures (Fig. 6, listed below each plot title). The first and most appropriate measure, given the rank-ordered nature of the quality labels, is the simple mean-absolute-error, or *distance error*, between the normalized prediction and the true quality. To provide an analogy to the supervised learning case with $\gamma = 0$, we also describe the *classification error*, wherein the prediction is rounded to the nearest integer and compared to the true label, indicating whether the algorithm predicts the right quality class. The state was provided by the 16 DNN features from the camera image, plus the photodiode data, resulting in a 19-dimensional state vector s. By using the features from the DNN representation-learning algorithm, we follow the architecture introduced in Fig. 2. The first three plots in Fig. 6 demonstrate the algorithm's capability to learn and generate the immediate process quality for the sensor data ($\gamma = 0$). The first plot in Fig. 6 illustrates the performance on a representative training process after multiple passes through the data. It can be seen that the algorithm is capable of learning the process and correctly predicts the true quality from the sensor data.

To also show the capability for predicting unknown processes, the algorithm was trained on one set of processes and then evaluated on a different, previously unseen process. The second plot in Fig. 6 shows prediction performance on the testing dataset after one training iteration through the training dataset, equivalent to online learning using the previous process as a starting point. After seeing only one pass through the training data the algorithm is already capable of following trends in the testing data, but the performance is still limited. While the learner is able to differentiate between the acceptable and bad parts of the process, the distinction between the good and bad classes is less clear, as can be seen by the distance error of 0.48. For comparison, learning was then continued iteratively on the training data until convergence, with testing again done on a single pass through the testing data. The results are shown in the third plot of Fig. 6. With added training the algorithm is able to clearly differentiate between the acceptable and the bad parts and the distance error has decreased to 0.28.[1]

While providing an instantaneous measure of quality is important, the use of a temporal-difference learning approach such as nexting provides advantages over conventional supervised learning in that it can also incrementally learn temporally abstract quantities—in effect, the performance over a given window of the future. This forecasting can be important for changing control parameters in advance of future observations. The last plot of Fig. 6 demonstrates the capability of temporal extended predictions. The algorithm was trained to anticipate the process behavior over a discounted window of approximately five frames ahead ($\gamma = 0.8$), which corresponds to 0.1 ms. As shown, the system is able to learn temporally abstract forecasts for the provided welding process, though the γ-normalized distance error increased slightly to 0.3. Classification error is not shown, as it has no clear semantics for $\gamma > 0$.

4.2. Control

To address the continuous action space inherent to the laser welding problem, we selected a continuous-action actor–critic reinforcement learning (ACRL) method, as outlined by Pilarski et al. [18,22], that uses a one-dimensional action space. This algorithm is depicted in Table 4, using the same conventions and notation as the nexting algorithm in Table 3. As detailed below, for this preliminary control experiment we used a simulated welding process. The current state is provided to the algorithm in form of the laser welding width; the output action of the learner is used to control the applied laser power, which is a real number in the range $[1, 5]$. The actor follows a stochastic control policy $\pi(a|s)$,

[1] The initial error at the beginning of the processes is due to the fact that the camera starts to record before the laser is switched on.

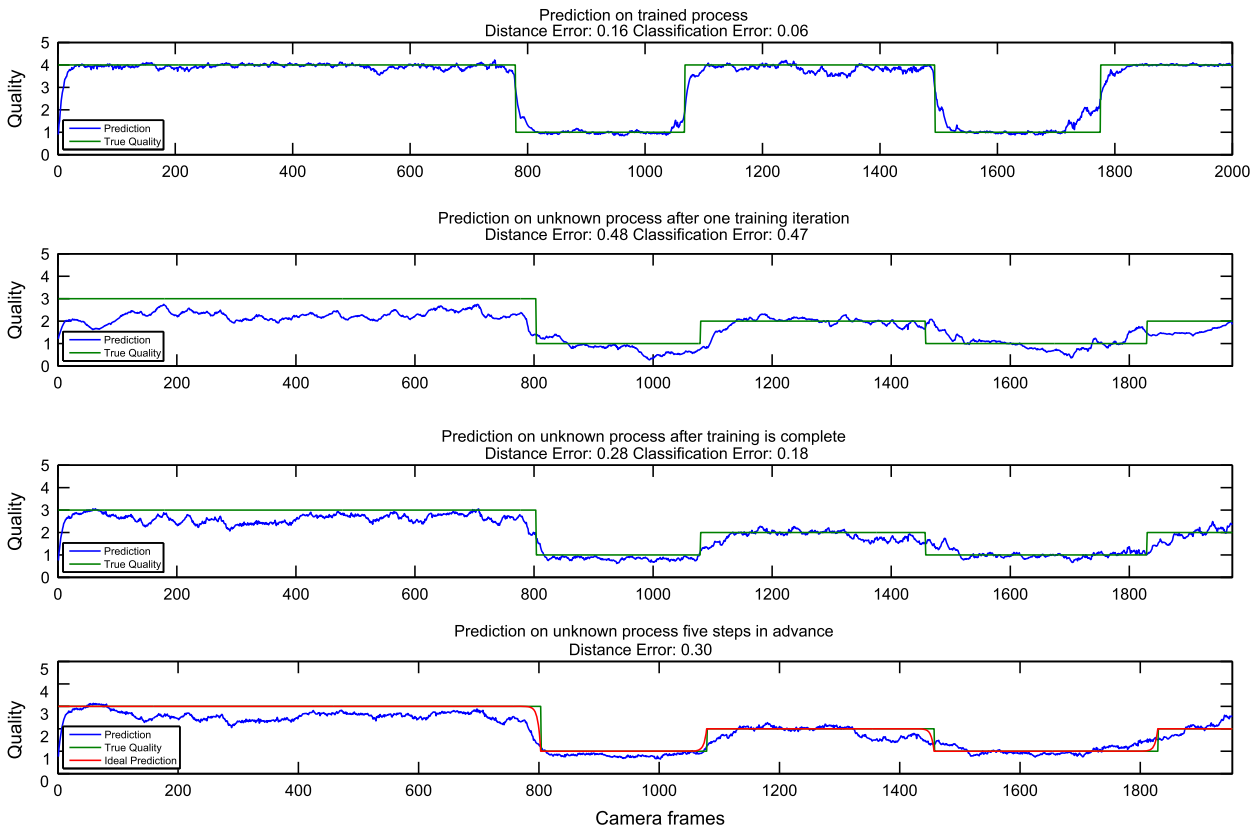

Fig. 6. Learning process for the nexting algorithm. The process is a contaminated weld of zinc-coated steel in an overlap position. The green line is the true process quality, estimated by an expert, while the blue line is the algorithm's prediction. The first plot shows the performance on a trained process. The second plot demonstrates the prediction on an unknown process, at the beginning of the training. The third plot is the prediction for the same unknown process, after the training is completed. The last plot demonstrates a γ-normalized temporally extended prediction, where each prediction is five steps in advance; to have a benchmark, the ideal γ-normalized prediction was calculated and is shown by the red line. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

sampled from a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with mean μ and standard deviation σ, which are linear combinations of a corresponding learned weight vector and the feature vector as shown in Table 4, lines 3 and 4. Actions were then mapped and clipped to the welding system's power range so that the applied power could not be bigger than the system's limits. The learned standard deviation is a measure for the exploration of the algorithm. If σ is large, the algorithm is more likely to explore different actions. As exploration steps usually result in a poor performance, the standard deviation has to decrease once the right mean has been found to achieve a stable, good performance. In a fully integrated system, the reward measure delivered to the control learning process would be a measure of the welding seam's integrity—i.e., the prediction of welding quality shown in the previous section and made available through nexting and learned representations. However, for rapid in-simulator testing of method viability, we provided the learner with a reward signal corresponding to the difference between the achieved welding depth d and a target depth d^* computed as $r = -0.5 + 1/[1 + exp(|d^* - d|)]$. This reward function was designed as a sigmoid function to keep the reward bounded and prevent large td-errors, which might lead to divergence. The function's range is $[-0.5, 0]$ with the value 0 corresponding to the correct welding depth.

In actor–critic learning, the actor learns a policy to generate actions while the critic evaluates the algorithm's performance so as to reduce the variance of the learning updates. For every sample quadruple (s, a, r, s') the critic calculates the td-error and uses it to update the value function v as well as the weights for calculating μ and σ. For our experiments we used the following learning parameters: learning rate $\alpha_v = \frac{0.1}{m}, \alpha_w = \frac{0.1}{m}$, discount rate $\gamma = 0.99$, trace decay parameters $\lambda_a = 0.3$, $\lambda_c = 0.3$ and sigma start $\sigma_c = 1$, where m is the number of active tiles. All trace and actor weight parameters w were initialized to zero. The algorithm was optimistically initialized, which means the weight vector v is set such that all states have an ideal value at the start of learning, i.e., zero.

For initial process control results, the control algorithm was tested on a preliminary simulator [41], which provides the welding seam depth based on the welding seam width, with a welding depth ranging from 10 mm to 20 mm and a welding seam width of 1.2–4.4 mm. The possible laser power ranges from 1.5 kW to 5 kW. To design the simulator closer to a real laser welding system, the subsequent changes were made following consultation with industry experts in laser welding. To model sensor noise, the welding seam width, which corresponds to the state of the system, was inflicted with white Gaussian noise of the standard deviation $\sigma = 0.05$. As real laser welding systems can make changes to their laser power only with a certain speed, the change in the simulator was restricted to 200 W per iteration. Finally, as real systems have physical response times, the welding was limited so as to only change by a maximum of 0.3 mm per iteration. This corresponds to 10% of the possible welding seam width range.

30 independent runs of learning using the simulator were performed and averaged. Fig. 7 shows the results for several actor–critic parameters over the learning process. As the algorithm selected its actions, i.e., the applied laser power, from a normal distribution, it performed random actions at the beginning of the learning process. In response to the received reward, the learner shifted the learned μ value towards the correct action and decreased σ to take less exploration steps as the learning process converged to a suitable target policy. We further see that with

Table 4
Actor–critic algorithm with continuous-valued output actions for controlling laser welding power.

Algorithm 2	Continuous-Action Actor–Critic Reinforcement Learning [18]
initialize: $w_\mu, w_\sigma, v, e_\mu, e_\sigma, e_v, s$	
repeat:	
$\mu \leftarrow w_\mu^\top x(s)$	// compute the current mean for the Gaussian distribution
$\sigma \leftarrow exp[w_\mu^\top x(s) + log(\sigma_c)]$	// compute the current std for the Gaussian distribution
$a \leftarrow \mathcal{N}(\mu, \sigma^2)$	// randomly choose action from Gaussian distribution
take action a, **observe** r, s'	
$\delta \leftarrow r + \gamma v^\top x(s') - v^\top x(s)$	// calculate the error, based on the current weights
$e_v \leftarrow \lambda_c e_v + x(s)$	// update the eligibility trace e_v, based on decay and visited state
$v \leftarrow v + \alpha_v \delta e_v$	// update the value function v, based on learning rate and error
$e_\mu \leftarrow \lambda_a e_\mu + (a - \mu)x(s)$	// update the eligibility trace e_μ, based on decay and visited state
$w_\mu \leftarrow w_\mu + \alpha_w \delta e_\mu$	// update the weight vector w_μ based on the td-error
$e_\sigma \leftarrow \lambda_a e_\sigma + \left[(a - \mu)^2/\sigma^2 - 1\right]x(s)$	// update the eligibility trace e_σ, based on decay and visited state
$w_\sigma \leftarrow w_\sigma + \alpha_w \delta e_\sigma$	// update the weight vector w_σ based on the td-error
$s \leftarrow s'$	// make successor state the current state

$\mu \in \mathbb{R}$:	Gaussian distribution mean	$\sigma \in \mathbb{R}$:	Gaussian distribution std
$\sigma_c \in \mathbb{R}$:	Starting value for σ	$a \in \mathbb{R}$:	Chosen action
$w_\mu, w_\sigma, v \in \mathbb{R}^n$:	Weight vector	$e_\mu, e_\sigma, e_v \in \mathbb{R}^n$:	Eligibility trace vectors
$s \in \mathcal{S} \subseteq \mathbb{R}^k$:	Current state vector	$s' \in \mathcal{S} \subseteq \mathbb{R}^k$:	Successor state vector
$r \in \mathbb{R}$:	Reward	$\gamma \in [0, 1]$:	Discount factor
$x(s) \in \mathbb{B}^n$:	Tile coded state vector	$\alpha_v, \alpha_w \in (0, 2)$:	Learning rates
$\lambda_a, \lambda_c \in [0, 1]$:	Eligibility trace decay factors		

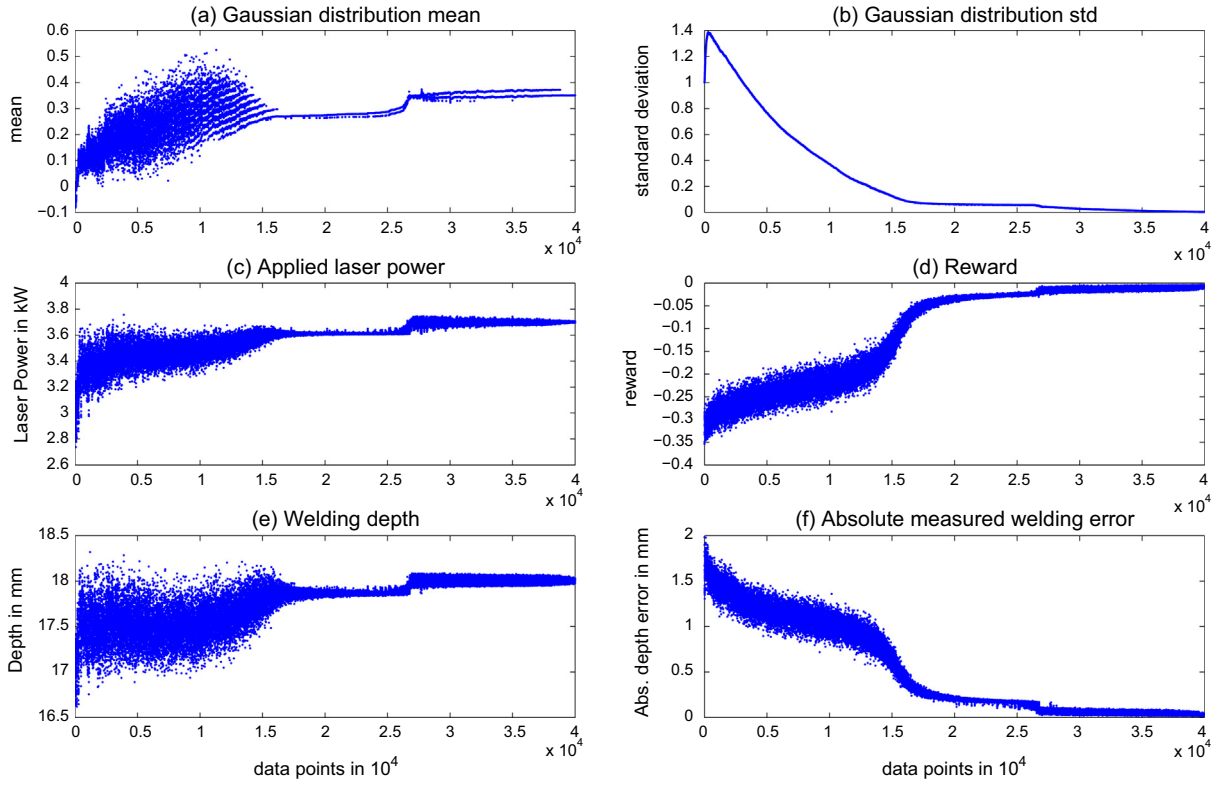

Fig. 7. Average learning performance for the continuous actor–critic algorithm over 30 independent runs. The two topmost plots (a) and (b) show the learned parameters for the Gaussian distribution, with its output actions shown in plot (c). Plot (d) shows how the algorithm performed over time in terms of reward. The plots (e) and (f) visualize the achieved welding depth and distance to the desired welding depth.

continued learning the system was able to precisely achieve the desired welding depth, as indicated by the depth error and the observed reward values both approaching zero. Although the algorithm had to cope with a noisy process and related limitations, it was able to find the correct solution with a precision that was only dependent on the amount of noise. None of the trials diverged, which further indicates the robustness of the algorithm.

5. Discussion

In this paper we proposed a new architecture that should be able to learn and control a wide class of specific laser-welding problems. In this architecture, representation learning was used to acquire salient features directly from welding process data. This learned representation was provided as direct input to a prediction

learner, such that the system was able to estimate missing information about the welding process—in this case, the quality of the welding seam, something that is impossible to measure directly during the act of welding. Finally, we described how knowledge in the form of features and quality information could potentially be passed to a system-integrated control learner to alter and optimize the power of laser welding in response to new situations and workpieces. Although the proposed architecture is flexible enough to deal with varying conditions in the process, the representation and knowledge framework in the present work has been specialized to deal with one specific laser-welding process of interest. Our architecture is capable of being extended as new process-related needs are identified, and we expect the present results will transfer well to different settings. Furthermore, while process quality was the target of nexting predictions in this work, in principle any process-related signal or group of signals could be predicted using general value functions and used to inform a downstream control process or control learner. Our proposed architecture provides an way to use more general predictive representations of state [42] in control learning.

Representation: In Section 3 it was shown that our DNN was able to extract meaningful features in a laser-welding setting, which could then be used to represent the current process state to integrated knowledge and control processes. Tests with a SVM classifier showed that the features generated by the DNN perform better than PCA features.

Prediction: The knowledge process demonstrated its capacity to learn about welding process behavior and not only to reproduce learned information, but also predict unknown processes. Fig. 6 demonstrates quality classification with 82% accuracy. If only used to differentiate between good and non-sufficient parts of the welding seam, the accuracy of the predictor is 93.1%. By also making temporally extended predictions, as in the bottom plot of Fig. 6, process behavior can be anticipated and control actions could potentially be applied in a preventative way before the quality changes. Computing predictions was done within 21 μs, which is within the requirements for real-time operation.

Control: The control algorithm described in Section 4.2 consistently converged to the correct solution in a short learning time. Even with distinct limitations, e.g., a noisy input and sluggish system response, the actor–critic approach was able to consistently converge to the correct solution. It was also fast enough from a computational point of view. The whole simulation took 66.3 s, whereas 52.6 s were used to simulate the laser welding system. Therefore, a single control iteration took only 0.34 ms, which is within the process requirements for real time capability (1 ms). The computer for the experiment used an Intel Core i5-2400 with a 3.1 GHz clock rate, 6 MB of shared L3 cache, 4 GB DDR3 RAM, and ran 64-bit Windows 7. The implementation of the simulator is applicable under the assumption that the representation algorithm is able to provide meaningful features about the systems state and that the prediction process can accurately predict the current performance. Additional work in the industrial setting is needed to demonstrate a fully integrated framework wherein learned DNN features and nexting predictions are used directly as input to the ACRL control learner during welding.

6. Conclusion

In this paper we have shown a new possible architecture for laser welding. Our proposed system includes methods to observe a process, build up knowledge about it, and then find ways to control it. The system has the capacity to improve its own performance, due to the way that it optimizes the process in terms of goals rather than in terms of mechanisms. It therefore promises to address key requirements of modern industry, in a way that our architecture combines fast learning with the capability to work well under changing circumstances.

In this work we described a possible combination of recent reinforcement learning and deep learning algorithms and provided insights into the impact this combination may have on laser-welding technology. To our knowledge, this is the first demonstrated combination of deep learning with nexting and general value functions, and one of only a very small number of papers describing the combination of reinforcement learning and deep learning systems, e.g., [8–10]. Additionally, it is the first demonstrated use of deep learning in laser welding and industrial production processes. This study is also unique in its use of reinforcement learning to acquire generalized predictions for use as inputs to a laser welding system. This makes the present work an important contribution to not only industrial process engineering, but also to the study of intelligent systems and machine intelligence. However, we do not wish to constrain our approach to the exact learning methods deployed in the present work; another important contribution of this paper is to suggest that the full integration of representation, prediction, and control learning into a single framework holds great promise for laser welding and other production engineering domains. We strongly believe that our approach is transferable to a broad range of industrial applications.

Acknowledgments

The project CCLW is performed within the framework of the European funding program Eurostars and is funded by the Federal Ministry of Education and Research – Germany. We also would like to thank all project partners, namely the Precitec GmbH & Co. KG, Germany and IREPA LASER, Illkirch, France, who provided the laser-welding data. The authors also acknowledge support from the Alberta Innovates Centre for Machine Learning and Alberta Innovates Technology Futures (Canada), the Natural Sciences and Engineering Research Council of Canada (NSERC), and thank Richard Sutton and Joseph Modayil for a number of helpful conversations.

References

[1] Jager M, Humbert S, Hamprecht F. Sputter tracking for the automatic monitoring of industrial laser-welding processes. IEEE Trans Indust Electron 2008;55(5):2177–84.
[2] Alippi C, Braione P, Piuri V, Scotti F. A methodological approach to multisensor classification for innovative laser material processing units. In: Proceedings of the 18th IEEE instrumentation and measurement technology conference (I2MC), vol. 3; 2001. p. 1762–7.
[3] Schroth G, Stork I, Wersborg G, Diepold K. A cognitive system for autonomous robotic welding. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS); 2009. p. 3148–53.
[4] Haykin S, Fatemi M, Setoodeh P, Xue Y. Cognitive control. Proc IEEE 2012;100 (12):3156–69.
[5] Zipkin P, S.S. of Management, M.I. of Technology. The limits of mass customization. MIT Sloan Manage Rev 2001.
[6] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th advances in neural information processing systems conference (NIPS); 2012. p. 1106–14.
[7] Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science 2006;313(5786):504–7.
[8] Lange S, Riedmiller M, Voigtlander A. Autonomous reinforcement learning on raw visual input data in a real world application. In: Proceedings of the 2012 international joint conference on neural networks (IJCNN); 2012. p. 1–8.
[9] Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, et al. Playing Atari with deep reinforcement learning. In: NIPS deep learning workshop 2013; 2013.
[10] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. Nature 2015;518 (7540):529–33.
[11] Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol P-A. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J Mach Learn Res 2010;11:3371–408.
[12] Clark A. Whatever next? Predictive brains, situated agents, and the future of cognitive science. Behav Brain Sci 2013;36:181–204.

[13] Liu F, Quek C, Ng GS. Neural network model for time series prediction by reinforcement learning. In: Proceedings of the 2005 IEEE international joint conference on neural networks (IJCNN), 2005, vol. 2; 2005. p. 809–14.
[14] Sutton RS, Modayil J, Delp M, Degris T, Pilarski PM, White A, et al. Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In: Proceedings of the 10th international conference on autonomous agents and multiagent systems (AAMAS), vol. 2; 2011. p. 761–8.
[15] Modayil J, White A, Pilarski PM, Sutton RS. Acquiring a broad range of empirical knowledge in real time by temporal-difference learning. In: Proceedings of the 2012 IEEE international conference on systems, man, and cybernetics (SMC); 2012. p. 1903–10.
[16] Modayil J, White A, Sutton RS. Multi-timescale nexting in a reinforcement learning robot. Adap Behav 2014;22(2):146–60.
[17] Degris T, Pilarski PM, Sutton RS. Model-free reinforcement learning with continuous action in practice. In: Proceedings of the American control conference (ACC); 2012. p. 2177–82.
[18] Pilarski PM, Dawson M, Degris T, Fahimi F, Carey J, Sutton RS. Online human training of a myoelectric prosthesis controller via actor–critic reinforcement learning. In: Proceedings of the 2011 IEEE international conference on rehabilitation robotics (ICORR); 2011. p. 1–7.
[19] Kimura H, Yamashita T, Kobayashi S. Reinforcement learning of walking behavior for a four-legged robot. In: Proceedings of the 40th IEEE conference on decision and control (CDC), vol. 1; 2001. p. 411–6.
[20] Thomas P, Branicky M, van den Bogert A, Jagodnik K. Application of the actor–critic architecture to functional electrical stimulation control of a human arm. In: Proceedings of the 21st innovative applications of artificial intelligence conference (IAAI), vol. 2009; 2009. p. 165–72.
[21] Peters J, Schaal S. Natural actor–critic. Neurocomputing 2008;71(79):1180–90.
[22] Pilarski PM, Dick T, Sutton RS. Real-time prediction learning for the simultaneous actuation of multiple prosthetic joints. In: Proceedings of the 13th IEEE rehabilitation robotics (ICORR); 2013. p. 1–8.
[23] Günther J, Pilarski PM, Helfrich G, Shen H, Diepold K. First steps towards an intelligent laser welding architecture using deep neural networks and reinforcement learning. Proc Technol 2014;15:474–83.
[24] Ancona A, Spagnolo V, Lugarà PM, Ferrara M. Optical sensor for real-time monitoring of CO_2 laser welding process. Appl Optics 2001;40(33):6019–25.
[25] Shao J, Yan Y. Review of techniques for on-line monitoring and inspection of laser welding. J Phys: Conf Ser 2005;15(1):101–7.
[26] Kroos J, Gratzke U, Vicanek M, Simon G. Dynamic behaviour of the keyhole in laser welding. J Phys D: Appl Phys 1993;26(3):481–6.
[27] Beersiek J. A CMOS camera as tool for process analysis not only for laser beam welding. In: Proceedings of the 20th international conference on applications of lasers & electro-optics (ICALEO); 2001. p. 1185–93.
[28] Sutton RS, Barto AG. Introduction to reinforcement learning. 1st ed. Cambridge (MA, USA): MIT Press; 1998.
[29] Serre T, Kreiman G, Kouh M, Cadieu C, Knoblich U, Poggio T. A quantitative theory of immediate visual recognition. Prog Brain Res 2007;165:33–56.
[30] Bengio Y, Lamblin P, Popovici D, Larochelle H, et al. Greedy layer-wise training of deep networks. Adv Neural Inf Process Syst 2007;19:153.
[31] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature 1986;323(6088):533–6.
[32] Moody J, Hanson S, Krogh A, Hertz JA. A simple weight decay can improve generalization. Adv Neural Inf Process Syst 1995;4:950–7.
[33] Vincent P, Larochelle H, Bengio Y, Manzagol P-A. Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on machine learning (ICML). ACM; 2008. p. 1096–103.
[34] Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. In: Proceedings of the 14th international conference on artificial intelligence and statistics (AISTATS), vol. 15; 2011. p. 315–23.
[35] Zeiler M, Ranzato M, Monga R, Mao M, Yang K, Le Q, et al. On rectified linear units for speech processing. In: Proceedings of the 2013 IEEE international conference on acoustics, speech and signal processing (ICASSP); 2013. p. 3517–21.
[36] Bengio Y. Practical recommendations for gradient-based training of deep architectures. In: Montavon G, Orr GB, Müller K-R, editors. Neural networks: tricks of the trade. Lecture notes in computer science, vol. 7700. Springer; 2012. p. 437–78.
[37] Bergstra J, Bengio Y. Random search for hyper-parameter optimization. J Mach Learn Res 2012;13:281–305.
[38] Cotter A, Shamir O, Srebro N, Sridharan K. Better mini-batch algorithms via accelerated gradient methods. In: Proceedings of the 24th advances in neural information processing systems (NIPS); 2011. p. 1647–55.
[39] Sutskever I, Martens J, Dahl G, Hinton G. On the importance of initialization and momentum in deep learning. In: Proceedings of the 30th international conference on machine learning (ICML), JMLR workshop and conference proceedings, vol. 28; 2013. p. 1139–47.
[40] Gilbert D. Stumbling on happiness. New York: Vintage; 2007.
[41] Min-Ying H, Shi-Junwei, Jinjin Li-Xin C, Yanqiu X. The research of welding parameters on weld shape in the laser deep penetration welding. In: Proceedings of the 2010 international conference on mechanic automation and control engineering (MACE); 2010. p. 3734–7.
[42] Littman ML, Sutton RS, Singh SP. Predictive representations of state. In: Proceedings of the 14th advances in neural information processing systems (NIPS), vol. 14; 2001. p. 1555–61.