Machine intelligence for adaptable closed loop and open loop production engineering systems

Johannes Günther

Technische Universität München
Lehrstuhl für Datenverarbeitung

Machine intelligence for adaptable closed loop and open loop production engineering systems

Johannes Günther

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzende(r): Prof. Dr. rer. nat. Oliver Hayden

Prüfer der Dissertation:

1. Prof. Dr.-Ing. Klaus Diepold
2. Prof. Patrick M. Pilarski, Ph.D., University of Alberta

Die Dissertation wurde am 20.09.2017 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 22.01.2018 angenommen.

Johannes Günther. *Machine intelligence for adaptable closed loop and open loop production engineering systems.* Dissertation, Technische Universität München, Munich, Germany, 2018.

Institute for Data Processing, Technische Universität München, 80290 München, Germany, `http://www.ldv.ei.tum.de`.

Acknowledgements

I would like to thank my supervisor Professor Diepold for giving me the amazing opportunity to perform research on my own. Especially as I am not an electrical engineer by training, he demonstrated trust in me and my abilities to learn, when deciding to let me work under his guidance.

I also want to thank Professor Richard Sutton and especially Professor Patrick Pilarski from the University of Edmonton. Not only did they invite me to visit for several months, but they provided invaluable help and enabled me to develop my own ideas within the field of applied reinforcement learning. Furthermore I want to thank Nadia Ady from the University of Alberta for proofreading and providing uncountable suggestions.

And last, but by no means least, I would like to thank my family for their never ending support, my friends for always having an open ear for all my problems during this time and my dog for distracting me from my work, whenever I needed it most.

Abstract

This thesis investigates the application of machine learning algorithms for industrial production engineering processes. There exist a wide range of applications, where traditional closed loop control approaches fail. In the setting of a closed loop control process, the controllers are usually strongly dependent on an *a priori* model and therefore cannot adapt to varying conditions and changing system dynamics. This might result in a poor control performance and a non sufficient disturbance rejection. There furthermore exist processes, where a closed loop control approach is not applicable at all, due to missing feedback. This is often the case for complex systems, where the process quality cannot be measured appropriately. This work will introduce new approaches addressing both problems.

In the first part, a traditional PID controller is enhanced by a general dynamic neural network to improve the disturbance rejection and control performance. It is evaluated using four typical control challenges. The control performance is tested with and without noise and a disturbance is applied. The results for this novel approach are compared to a standard PID controller and to one state of the art model-based control approach.

The adaptive PID controller proves to be superior to the standard PID controller for all conducted experiments. It furthermore outperforms model-based approaches in 13 out of 16 comparisons. The results indicate this approach to be robust and well suited for problems with changing dynamics, as the new approach can continue to learn, while running.

The second part takes industrial laser welding as an example of a complex process that can neither be modeled nor sufficiently observed, resulting in an open loop process. To overcome this shortcoming, a new intelligent architecture is introduced, combining deep learning neural networks and general value functions to provide a quality signal. Using this feedback, closed loop control via an actor-critic reinforcement learner is enabled.

The newly combined architecture demonstrates that a learning approach cannot only provide a certain degree of invariant perception of the laser welding process but furthermore can reliably extract a quality signal out of the sensor data. This allows closed loop control. The appropriate control signals are learned and provided by the actor-critic algorithm.

Together, the two parts in this work demonstrate how modern industrial processes can benefit from the application of machine learning algorithms. As the approaches are generic, they are not limited to the particular examples but can be applied to a wide range of industrial processes. Furthermore, the machine learning algorithms introduced in this thesis can continue to learn and adapt, yielding great potential for autonomous systems.

Contents

List of Figures

List of Tables

1 Introduction

This chapter will explain the motivation of this work by introducing the general research problem, showing already existing approaches and solutions as well as their limitations. Furthermore the implications in terms of open questions and possible improvements will be presented and the structure of this work will be outlined.

1.1 Problem statement

Labour was the first price, the original purchase-money that was paid for all things. It was not by gold or by silver, but by labour, that all the wealth of the world was originally purchased.
-Adam Smith, The Wealth of Nations Book I, Chapter 1

When considering the cost of any good, the largest portion will typically be the labour necessary to create it. During the last centuries, numerous efforts have been made to reduce the (manual) labour required by production processes. Specialization and automation during the first industrial revolution not only dramatically increased productivity but at the same time reduced the price for goods, thus making them available to a broader circle of customers. The development of production via automation is still ongoing. The introduction of technologies like new powering methods and the capability of running processes without the need for human supervision were significant breakthroughs for industrial processes.

To remove the need for human supervision from automated processes, we need an approach called closed loop control. In a closed loop system, the system output, in the form of a feedback signal, is fed back in a loop to improve the control performance; therefore, these systems are also called feedback systems. The feedback signal is usually provided by an appropriate sensor, sampling the system output. One of the first sensors to be used for a closed loop control system was a centrifugal governor [1], a sensor based on the simple physical relation between angular speed and centrifugal force.

The actual output is compared to the set point, which is the desired output, and the difference is the control error. As the controller receives this error as input, it has means to react to the actual system output and provide the correct control signal. Unfortunately, the output of a system cannot always be measured, due to a lack of sufficient sensors. In this case, an open loop control approach is applied. Here, the controller provides a control signal to the system, but without feedback it cannot evaluate the effect of the provided signal. Such production processes necessitate either a very costly setup, requiring huge amounts of time and manual labour, or a costly quality inspection afterwards—or both.

Even with an adequate feedback signal, a closed loop control approach might not prove sufficient. If unforeseen disturbances occur or the applied controller fails to account for the complexity of the system to be controlled, it might still fail to achieve a sufficient control performance.

An ideal control system would not rely on an *a priori* model or any assumptions about the control system, but independently identify the system dynamics, learn them and consistently adapt its understanding of the system dynamics to be controlled. Another important feature for control systems is the disturbance rejection. This defines the ability to identify and counter disturbances in the system as they occur. Furthermore, the control system needs to be able to change its own parametrization to adapt the control output to the requirements of the control system. Such a control approach would ideally neither require human assistance during the setup process nor during its control performance.

Increasing demand for customization and shorter product life cycles have motivated completely new approaches for process control, namely data-driven and connected ones. By combining physical systems with modern computer science, a connected and communicating production environment, called “Industrie 4.0” was introduced. This environment includes modular programming with the actual hardware in order to easily change production processes, based on individual requirements. As the number of connected processes increases, so does the available data. Collecting and analyzing the data is a first step towards improving production processes in a data-driven way. Making use of the collected data is the second step in order to not only learn from processes in the aftermath, but to use this data to improve processes while they are running. A way to not only monitor, but learn from data and apply the learned information for improvement is machine learning. Machine learning algorithms yield great potential to automatizing processes [2]. However, determining the most beneficial way to apply them is still ongoing research.

This thesis provides suggestions for two different philosophies of applying machine learning algorithms for industrial production systems to generate flexible and self-learning systems. Such systems promise to result in further decrease of manual labour and costs.

1.2 State of the art and limitations of existing approaches

The following section will provide an overview of traditional control algorithms for both control approaches, briefly explain their main principles and also highlight their shortcomings.

1.2.1 Closed loop control systems

For simple feedback control tasks, numerous controllers have been invented and applied. One of the first controllers to be used for closed loop control was the proportional integral differential controller (PID controller) [3]. For a detailed introduction to PID control, see Section 2.2.1. A PID controller calculates its control output based on the current error, the error derivative and the error, integrated over time. Each measure is multiplied with a constant and then summed up. The constants are chosen based on the system. The PID controller is one of the most widespread controllers, likely due to its simplicity.

Regrettably, PID controllers are limited when dealing with disturbances or complex systems. As the PID controller is inherently based on a linear model, it is mostly applicable to linear systems [3]. Nonlinear systems have to be linearized around the set point. This procedure results in a susceptibility to disturbances [4]. A big enough disturbance will force the PID controller out of the range in which the assumption of linearity holds. Outside of this range, the initially chosen constants are no longer correct for the system's behaviour. A way to compensate up to a certain degree for such a disturbance is to increase the gain. The term "gain" refers to the constant multiplied with the current error in a PID controller, and to the analogue in other controllers, and so determines the controller's response to the current control error. A controller with a high gain will react with a high controller output in order to quickly reduce the control error. This will push the controller back towards the linearized range. With a high enough gain, the effects of other parameters will be negligible and the disturbances can be controlled. Unfortunately, a high gain can also result in overshooting and oscillations, which might lead to instability, thus resulting in a failed control approach.

To achieve both disturbance rejection and stability, robust control approaches were introduced [5, 6]. There exist a significant number of different approaches for robust control, such as H_∞ controllers [7, 8, 9], model-based controllers [10, 11, 12] or controllers based on Lyapunov equations [13, 14, 15]. Robust control designs have successfully been applied to a wide range of control problems, including UAVs [16], wind turbines [17], thermodynamic problems [18], smart grids [19] and electric circuits [20]. Robust control relies on initial parametrization of the controller based on *a priori* models. Because of this dependency on *a priori* models and failure to adapt to operational conditions, robust control is still susceptible to significant changes in the control system and to changing and unforeseen conditions.

An extension of robust control is adaptive control, where the controller tries to adapt to changing conditions [21]. Adaptive control can be categorized into four different classes [22].

The first approach to adaptive control is gain scheduling. Here, an auxiliary variable is introduced that is highly correlated with changes in the system dynamics. The controller gain is then adapted based on the auxiliary variable. Gain scheduling is very popular in practice and has been applied to a variety of different problems [23, 24, 25]. However, the performance of gain scheduling is mainly dependent on the correlation of the auxiliary variable with the process. As this dependency has no feedback loop, it is structured in an open loop. If the system dynamics change, the correlation of the auxiliary variable with the process might decrease, resulting in inappropriate gain scheduling [22].

The second type of adaptive control is based on reference models. Model-based adaptive control can be implemented using either a series high-gain scheme or a parallel scheme [22]. The series high-gain scheme was originally introduced for applications in flight control. Like the simple high-gain scheme described above, the series high-gain scheme for model-based adaptive control uses high gain to rapidly achieve a desired set-point. The utilized reference model is designed to provide the desired output, i.e. the set-point, in response to an input. Due to the high gain, the system is forced to follow the reference model's output [26, 27, 28, 29]. This type of control does not account for circumstances unforeseen by the designer of the reference model. Ideally, the gain is set as high as possible, but decreased whenever instability is sensed. Although being quite intuitive, this scheme has the drawback of consistent oscillations. Furthermore, the system becomes unstable when the reference inputs enter saturation, due to no longer being able to sense the increasing oscillations [22].

In the second model-based approach, the adaptive controller is organized in a parallel scheme. Again, the model specifies the desired system output. The typical scheme consists of two loops, where the inner loop resembles a classical closed loop setting and the outer loop adjusts the controller parameters. The outer loop tries to adjust the parameters in a way that the system output matches the model output. Early parallel adaptive control schemes were lacking stability proofs that could only be provided by applying Lyapunov theory as of 1966 [30]. Parallel model-based adaptive control schemes have been successfully applied to different control tasks, including robotics [31], traffic [32] and aircraft [33]. The adaptive control approaches introduced so far follow a direct approach in the sense that they directly update the controller parameters based on the system behaviour.

The third method for adaptive control uses self-tuning regulator algorithms. Self-tuning regulators have two sets of parameters, one for the controller and one for the system. The system parameters are initialized based on a control design for a known system that is similar to the system at hand. As these parameters might not be correct, they are refined by a procedure known as a recursive system identification algorithm. The estimated system parameters, which are assumed to be true, are then used to compute the controller parameters. This procedure follows the certainty equivalence principle [22]. The process of system identification and parameter computation is ongoing. Self-tuning regulators were originally introduced by Kalman [34]. The scheme consists of two loops, one for the control and one for the system identification. This setup is very flexible in terms of the utilized algorithms, as the specific controller and the system identification can be performed by any given capable algorithm. However, the control will eventually only perform as well as the system identification. Additionally, the stability and convergence analysis turns out to be nontrivial due to the usually nonlinear transformation from the system identification to the controller parameters. Self-tuning adaptive algorithms have been successfully applied to numerous problems [35, 36, 37]. In contrast to the direct adaptive control approaches, where the the controller followed the provided model, here the adaption is done indirectly by first identifying the system and then adapting the controller. System identification and control are therefore separated.

The three previously introduced approaches were based on heuristics. While those heuristic approaches do perform well, they are usually mainly dependent on the provided model. The last class of adaptive controllers are stochastic control approaches. A stochastic approach could be seen as a unified adaptive learning framework, as it includes the indirect adaptive control approach using system identification with the direct, where the controller parameters are directly updated based on the systems output [22].

In stochastic adaptive control, the system is represented by a stochastic model. This stochastic model uses a state estimator, which learns a mapping from controller output and system output to appropriate controller parameters. The performance of the stochastic model is measured against an optimization criterion and improved based on a loss function. Depending on the optimization approach, the typical exploration-exploitation dilemma [38] can occur: While the controller does need to experience new outputs in order to learn about them, the control performance will usually drop, as exploration might mean to deviate from the set point. If the controller only tries to achieve the best possible control performance, based on its current information about the system, it might not learn enough about the control problem at hand. Additionally, the convergence and stability for this method cannot be formulated in general [39, 40, 41], as they depend on the choice of stochastic approach. These limitations lead to the field of stochastic adaptive control not being as well investigated as the other adaptive control approaches.

With the exception of stochastic adaptive control, a general drawback of adaptive control is the need for an *a priori* model. The performance of any of these adaptive control approach is limited by the applicability of the employed model, limiting the ability to learning and adapting to completely new and unforeseen changes in the process. Traditional adaptive control is therefore only usable within a narrow range, where the disturbances are limited and the general system dynamics do not change significantly. Successfully applied control approaches like Backstepping, LQ-regulators or Smith predictors suffer from these problems and are prone to these difficulties [42].

The effect of unexpected influences and changing or uncertain systems on the process is in fact a big challenge in control theory and by far not solved for the possible variety of possible disturbances. Instead of attempting to create a complex *a priori* model that includes all possible disturbances and system dynamic changes, this work follows the idea of data-driven learning. By applying modern machine learning algorithms it should be possible to learn and react purely based on sensory data and therefore without the assistance of human beings, both for controller design and operation. Using machine learning in such a data-driven way to consistently learn and update the implicit model for the system dynamics is closely related to the approach of stochastic adaptive control. If successfully applied to different control tasks, a machine learning based autonomous adaptive control approach can be seen as a further unified version of the previously introduced stochastic adaptive control approach. The ability for a controller to consistently improve its own performance would be an important stepping stone towards self-learning and self-improving processes that might help to fulfill modern production requirements [43].

Introducing such new controllers into production systems to replace controllers that are already in-place is expensive. To apply model-based controllers, the system has to be analyzed to find a model for the controller and the new control approach has to be implemented—either in software or in hardware. Another option is to modify the already in-place controllers to extend their capabilities. Machine learning can be used in a closed loop setting in a very intuitive way. It can be integrated into the already in-place controller with minimal effort and only be provided with the measurements that were already used for the controller so far. This approach has the advantage of being applicable with only minimal changes to the set up. It can therefore be used in almost every production system where the special capabilities of machine learning promise a benefit. Such a machine learning extended control approach can be realized using microcontrollers or other small and inexpensive computational units, minimizing the cost.

In this thesis, an approach to extend existing systems and hardware by machine learning is shown by using general dynamic neural networks for online PID tuning. Neural networks are a type machine learning algorithms and they will be introduced in detail in Section 2.4. Even without changing the general setup, by integrating a general dynamic neural network into the regular PID control framework, the PID controller parameters can constantly be tuned, ensuring an ideal parametrization at all time.

Neural networks have already been introduced to PID parameter tuning in three different ways. The different approaches mainly vary in the specific type of neural network to be included into the closed loop control approach.

The very first approach to include neural networks into a closed loop control setting was utilized simple feed-forward neural networks [44, 45, 46]. Such neural networks are deterministic after training such that the same input will always result in the same output. While they can learn highly nonlinear mappings from input to output and represent any possible function, they are not well suited for time varying problems due to their insensitivity towards temporal structure.

To provide such a temporal sensitivity, external dynamics have been included into the neural network PID tuning approach. Additional information about the system's previous states and outputs is externally stored and provided to the neural network via additional inputs. As expected, the performance for parameter tuning is improved by the addition of temporal information [47], but it comes at a price, but comes at a price. The additional inputs enlarge the neural network, making it not only more expensive to compute, but they also increase the requirements for memory. Furthermore, bigger neural networks tend to get stuck in local minima, rather than converging to the global minimum, resulting in a diminished performance [48].

To keep the neural networks as small as possible but still make use of temporal information, neural networks where temporal information can be stored within the neural network were evaluated for their performance on PID parameter tuning. In a first step, recurrent neural networks were utilized. The term "recurrent neural networks" comes from the recurrent connections of specific neurons, i.e. the neuron passes its output to itself for the next temporal step. This way, previous inputs are stored within the neural network and influence the mapping of current inputs. These recurrent neural networks have demonstrated a superior performance for linear [49] and coupled linear benchmark systems [50]. They also show the capability to reject small disturbances significantly better than standard PID controllers [49, 51]. To successfully control single-input single-output nonlinear systems, diagonal [52] and quasi diagonal recurrent neural networks have been used [53]. However, all these approaches introduce strong restrictions onto the structure of the neural networks. They were specifically designed for the given control by hand, resulting in significant manual labour during the design process. Furthermore, all neural network based PID tuners were only applied to specific control tasks that do not resemble the full spectrum of control challenges. Eventually. all the proposed papers do only evaluate the performance of the PID tuning approach via neural networks in a qualitative way, not using typical numerical measure.

1.2.2 Open loop control systems

In the previous section, only closed loop systems were considered. Open loop control problems require very different control approaches. While some open loop control problems can be run without feedback, more complex systems require feedback but have output that can only be described by multimodal and multidimensional sensor signals. Such signals cannot feasibly be used as direct input to a controller, but motivate data-driven approaches for control. In this thesis, an industrial laser welding setting is investigated as a representative system of this type. Laser welding is introduced in more detail in Section 2.1. Numerous techniques have been introduced, for laser welding specifically, to fuse sensor signals and compress information into lower dimensions and extract important underlying structures, e.g. principal component analysis (PCA) [54], linear discriminant analysis (LDA) [55] or ISOMAP [56]. However, most of these techniques have problems dealing with variance within the original data, rendering them almost useless for processes where variance might be introduced due to environmental disturbances or changes in the sensor setup.

The inability to evaluate the current process quality at any time is another huge constraint when open loop control is applied. There are several approaches to address this issue for laser welding, e.g., *a priori* models [57], envelope curves [58, 59] or look-up tables [60]. However these techniques are restricted either in applicability (*a priori* models), accuracy (envelope curves) or scalability (look-up tables). A way to overcome those problems is to use machine-made predictions about the processes, but most of these approaches are dominated by linear models, which often fail to account for the complexity of nonlinear and time-varying problems [61]. These techniques are also limited in their ability to adapt to changing process parameters or are not even capable of adapting at all.

To develop techniques which adapt over time and are furthermore not necessarily limited to a priori models, machine learning can be used to learn from the available sensor data. In a first machine learning approach for laser welding, a principal component analysis (PCA) and a support vector machine (SVM) were combined [62]. While the combination of these two algorithms is able to adjust the laser power for a process with a changing laser welding velocity, it is only valid for a defined setup. Such a closely controlled setup is not reasonable in a real production environment. With even small deviations in the sensor setup, the representation provided by the PCA is rendered useless. Furthermore, a human expert is needed every time the process is set up to train and evaluate the PCA and SVM.

1.3 Formulation of the research problem

Classical model-based control approaches cannot sufficiently solve the problems created by increasingly complex systems and new demands for flexible and adaptive processes. As a result, new ways to deal with these problems have to be found. A promising approach is data-driven closed loop control. This thesis will evaluate two different methods to apply data-driven closed loop control to solve the upcoming questions. The first method is to evaluate the extension of already in-place closed loop controllers by machine learning to answer the following questions:

- Can a data-driven machine learning approach, based on general dynamic neural networks, be applied to create a unified stochastic approach for adaptive control for PID control?

- How well does this novel approach perform in terms of convergence and disturbance rejection, when compared to standard PID controllers and state of the art adaptive control algorithms?

More complex systems might result in a problem, where the systems output in terms of process quality is not easily measurable. In such a setting, a closed loop control approach is not naturally applicable, but still desirable. Industrial laser welding is such a process and will be employed as a use case. For this open loop control problem a second machine learning approach is utilized to answer the following questions:

- Can a combination of machine learning algorithms be applied in a way to enable a closed loop control approach for industrial laser welding, as a representative for complex industrial production processes without appropriate feedback signals?
- How can different machine learning algorithms be combined in a useful way to solve complex problems and which algorithms serve this architecture?
- How can we use machine learning to solve control problems without feedback signals?

If all these questions can be answered in a positively way, the answers can lead to an extension of existing control tasks. Additionally, they open the possibility of applying closed loop control to production processes that so far have to be run in an open loop setting, consequently of requiring extensive manual labour to evaluate the processes' performance afterwards.

1.4 Contributions

This thesis motivates and answers the formulated research questions.

I define the requirements for industrial process control to enable learning and adaptation during production. For closed loop control systems, I describe the most common controllers, like PID control, robust control and adaptive control. All of these control approaches are evaluated on whether they meet the requirements and their shortcomings are explained in relation to the required properties. I then proceed to introduce existing neural network based PID control approaches. To improve these existing approaches and evaluate them properly, I introduce a stochastic adaptive control approach, based on neural networks with arbitrary connections and delays.

To evaluate the new suggested control approach, I take an empirical approach using four different simulated test systems chosen as representative of the most common challenges in control theory. I implement a standard PID controller and, for each test system, one state of the art controller from either the family of adaptive control or the family of predictive control. The performance of these controllers is compared with the new approach through evaluation in the different test systems, including simulations with noise and disturbances, and compared to the state of the art algorithms.

For the open loop control systems, I use an industrial laser welding system as a suitable representative embodying the most typical challenges. I introduce the common open loop industrial procedures and a new machine learning based closed loop approach and explain their shortcomings, compared to the formulated required capabilities of a suiting controller.

To enable a machine learning based closed loop control approach for laser welding, I collect a sufficient laser welding data set in the laboratory, that consists of 100 individual laser welds. I then created a novel architecture that employs machine learning algorithms from the fields of deep learning and reinforcement learning to resemble basic human capabilities. This architecture perceives the laser welding process and extracts information about the current laser welding quality, allowing a closed loop control approach. For all used machine learning algorithms, an extensive parameter search is conducted and their final performances is evaluated, using the typical machine learning error measurements, e.g. F-scores. To evaluate a reinforcement learning based control algorithm, I program a laser welding simulator, including suggestions and feedback from industrial laser welding experts. The control algorithm proved to be able to successfully find the correct solution and achieve a steady laser welding depth, thus closing the loop for an automated laser welding system.

1.5 Scientific approach and structure of this work

Together, the information in the following chapter will provide the basic knowledge to follow the later introduced ideas on applied machine learning for production processes. The following chapter will provide a comprehensive introduction to the industrial process of laser welding, as this process will be considered as an example of a complex and hard to control industrial open loop control system. In addition, the idea of closed loop control in general and PID control in particular will be provided, followed by the introduction of the applied machine learning algorithms.

In the third chapter, the classic case of closed loop control via PID controller will be considered and enhanced. The general setting will be a control task where the system's output can easily be observed via sensors and therefore a closed loop PID control can be applied. Within this thesis, PID controllers will be extended by a general dynamic neural network that has access to the PID parameters in order to adjust them while the control process is running. All four different evaluation systems, namely the non-linear two-tank system, the inverted pendulum on a cart, the system with a non-neglectable time delay and the chaotic fluid system, will be implemented, using their respective differential equations. Each system will be implemented with and without noise and the algorithms will be tested for robustness, using disturbances that typically can occur in each system. To ensure independence from pseudo random numbers, all results will be averaged over 30 independent runs.

The fourth chapter will address the challenge of complex systems, whose system output is not measurable in terms of quality and therefore does not allow closed loop control without further effort. The laser welding process will be used as an example and real laser welding data from different laboratories will be investigated. Different machine learning algorithms will be implemented and applied to the data to generate an artificial systems output which can then be used in a closed loop way. The suggested combination of algorithms follows an architecture that combines them in a way to resemble an intelligence inspired way of processing information in three steps—representation, prediction and control. Each step will be individually explained in detail and also evaluated. For the representation, two different deep learning approaches will be introduced and compared. The prediction algorithm will be tested, using a leave one out cross-validation to evaluate its performance. Lastly the control algorithm will be evaluated, using an appropriate laser welding simulator.

The fifth chapter will critically discuss the results and the thesis will be concluded in the following sixth chapter.

2 Theoretical background

This chapter introduces the industrial process of laser welding, which will serve as a use case. The general setup and the specific challenges that arise from laser welding will be explained, as well as the sensor setting used in the experiments. Furthermore, the foundations of control theory in general and the PID controller in special will be introduced. As this work suggests the application of machine learning algorithms for control problems, machine learning algorithms that will be used in this thesis are explained in this chapter in general and the applied algorithms in more detail. This will include neural networks and reinforcement learning.

2.1 Laser welding

Joining technologies are among the most important manufacturing techniques in industry. Among various joining techniques, laser welding combines the highest precision with a fast joining speed [63]. An example of a laser welding configuration can be seen in Figure 2.1. The laser is fed through a fiberglass cable, passes the collimation lens and is focused on the welding area by the focus lens. Where the laser hits the workpiece, the material is heated until it vaporizes. As the vaporizing metal expands, a tunnel or the so called "keyhole", is created. This keyhole remains open for as long as the process is running. Laser welding achieves its unique precision through a very small heat-inflicted zone around the keyhole.

The advantages of this particular joining technique are paid for with certain limitations and problems. The process is too complex and noisy to be accurately modeled by a mathematical model and therefore not controllable by any model-based control approach [64]. The current best solution is to closely and consistently monitor the process [65]. A successful approach for closed loop control has not been established, at this point.

In Figure 2.2, the main problem that arises for control purposes is illustrated. Shown are top views and cross-sections for an overlap welding configuration, which means the two workpieces to be combined are lying on top of each other. The figure shows an ideal weld on the left side and a completely insufficient weld on the right. From the top view it is hard to differentiate the qualities—only the cross-section reveals the true weld quality. The two metal plates are not connected at all. However, humans, even when seeing only the top view, can use the smoothness of the welding seam to distinguish the difference. This assessment is not trivial for an automated control approach and cannot be done, so far.

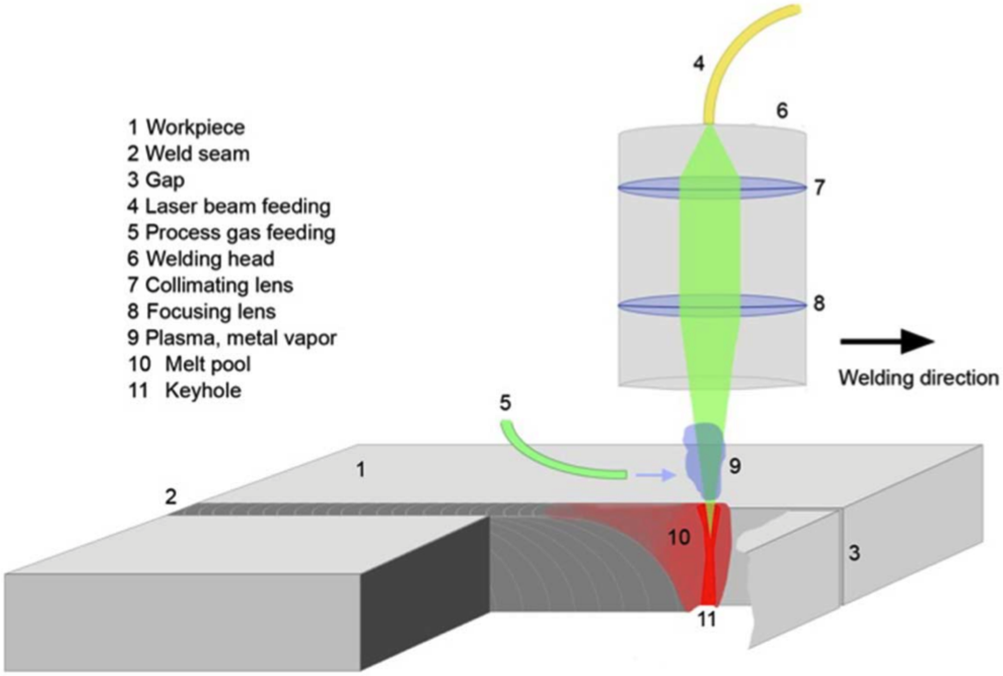


Figure 2.1: Example laser welding setup with a butt joint configuration. Courtesy of Stefan Müller-Meerkatz.

This is one example of the difficulties arising when the laser welding process is only controlled and observed by a camera system. Recent work has shown a cognitive approach that performs well for a defined and pretrained workpiece in a lab environment [66]. In real production systems these preconditions can not be fulfilled. Besides changing environmental conditions, like variations in temperature, humidity or the welding gas quality, there might also be variations in the workpiece itself. These contain, but are not limited to, changes in the chemical composition, the thickness of the workpiece and contamination of the surface [67].

A very common process monitoring setting—which was also used for the experiments described in Chapter 4—is to observe the process using a camera-based system in combination with photodiodes [68]. While the camera provides important geometrical information about the size and shape of the keyhole [69] and its surrounding heat affected area [70], the photodiodes monitor the process at different wavelengths in order to provide information about the process temperature (1100–1800nm), the plasma radiation (400–600nm) and the laser back reflection (1050–1080nm). A typical frequency with which the keyhole oscillates is around 500Hz [71] and therefore all sensors have to sample with at least twice this frequency. In the monitoring devices at hand, the camera samples with a frequency of 1500Hz and a resolution of 144 × 176 pixels, while the photodiodes sample with up to 40kHz.

Usually, the control of the robot arm that moves the laser and the control for the laser itself are separated. For a potential control algorithm to be inexpensively applied to such an existing setup, it is natural for the control algorithm to only influence the system through changes to the laser power.

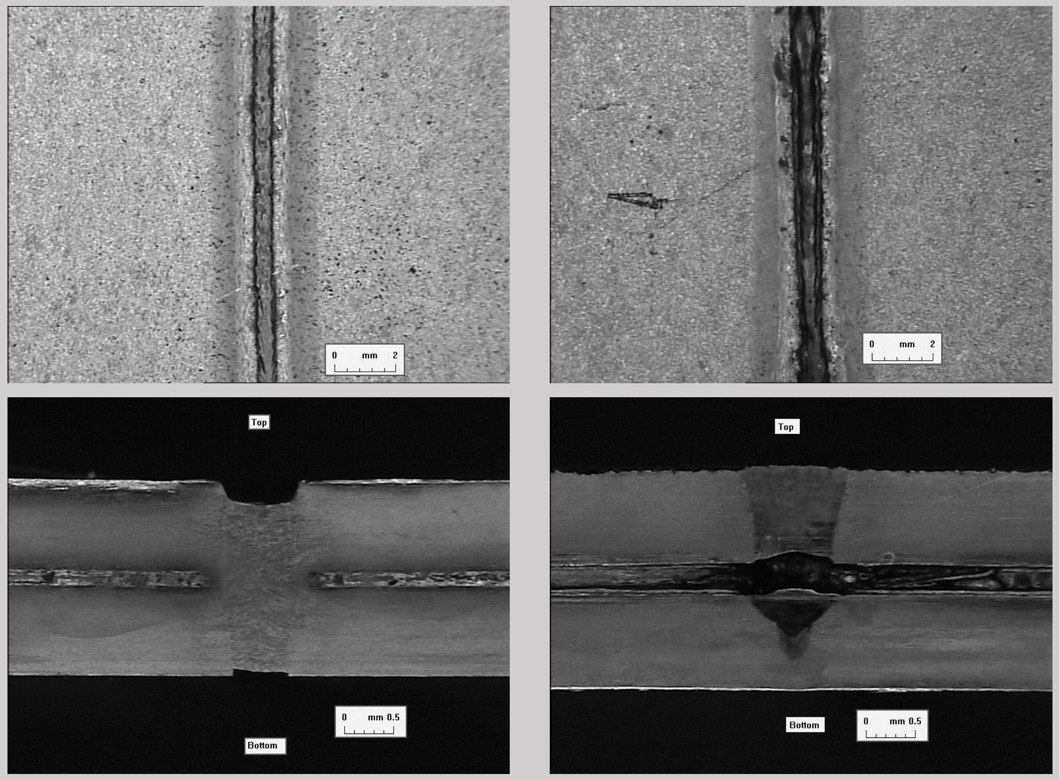


Figure 2.2: Laser welds and corresponding cross-sections for zinc-coated steel in overlap position. [72]

2.2 Control theory

Control theory, in its widest definition, deals with the behavior of dynamic systems, dependent on their inputs. It can be divided into two main approaches—open loop control and closed loop control. In an open loop control setting, the controller receives a set point signal and calculates and passes its input into the system (plant) in order to trigger a certain system output. The main problem with open loop control is that for this approach to work well, the system has to behave exactly as the controller anticipates. Any unforeseen system behaviour or external influences will lead to the control approach failing. As this assumption does not hold for most control problems, especially in the manufacturing domain, the system's output is usually used in a closed loop form, shown in Figure 2.3. By subtracting the system's output z from the set point z^*, the control error e is calculated and fed back into the controller. As this setup allows the controller to react to the actual system output, a closed loop control usually leads to better disturbance rejection, stabilization of unstable processes, improved tracking of the set point and less sensitivity to parameter variations. A more detailed introduction into control theory can be found in [73, 74, 75].

The most simple form of control problems are linear and time-invariant systems. When deciding upon a controller for such a problem or on the parametrization of the controller, either the knowledge of the systems transfer function or a state-space representation is required. The very basic idea is to create a mapping of the form

$$z = T(z^*), \tag{2.1}$$

where z is the system's output, z^* is the input and T is the transfer function. Using this approach, the complete system can be checked for observability and controllability. While observability describes whether each internal state can be observed through the system output, controllability further indicates whether it is possible to force the system into each possible internal state via the input signal. If a system is completely observable and completely controllable it is a minimal or stable system [76].

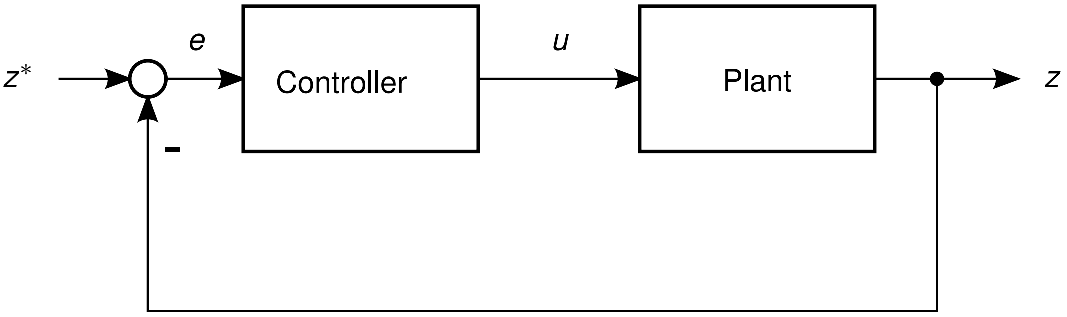


Figure 2.3: Closed loop control setting with a negative feedback loop.

2.2.1 PID control

The Proportional-Integral-Derivative controller (PID controller) is one of the most basic and most widely used controllers. There are over 75 patents on PID tuning in the world and more than 45 commercial PID software packages [3]. The idea of not only using the current control error but also its derivative and past control error comes from observing helmsmen trying to keep ships on course [77]. This control approach can be mathematically explained by the equation

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{d}{dt} e(t), \tag{2.2}$$

where $u(t)$ is the output of the controller, $e(t)$ is the control error and K_P, K_D and K_I are the constants for the proportional, differential and integral parts, while τ is the variable for the integration. A graphical interpretation of this dependency in the Laplace domain is depicted in Figure 2.4. The different constants can be seen as different filters to reduce the steady-state error. The proportional term serves as an all-pass, while the integral term and the differential term compensate for low frequencies and high frequencies, respectively [3]. Choosing these three parameters is referred to as tuning the controller. There are different measurements to evaluate the controller performance, namely: rise time, overshoot, settling time, steady-state error and stability. Each measurement is differently influenced by changing the parameters, so finding the best combination is the most important task when implementing PID controllers. An extensive introduction on PID control system analysis can be found in [3]. As the PID controller has only three variables, its performance for complex systems is limited. Especially for nonlinear problems, there does not exist a single set of parameters that suffice to cover the control needs for every possible state.

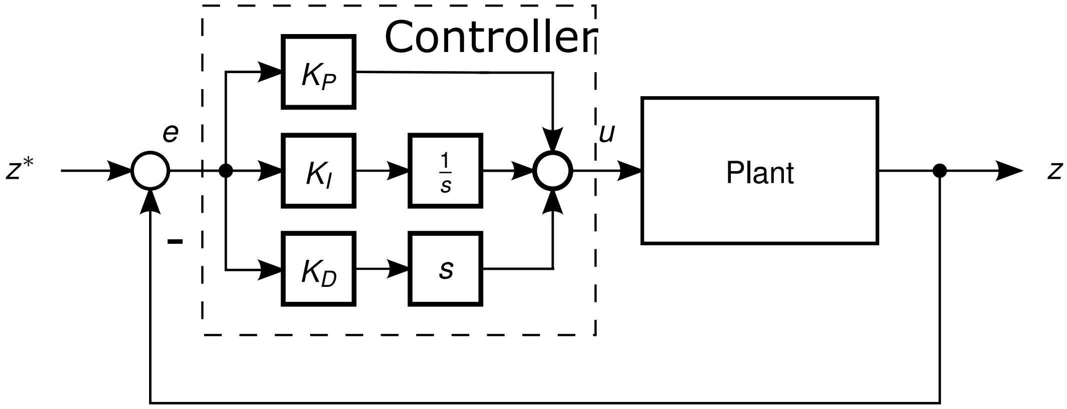


Figure 2.4: A closed loop control setting, using a PID controller.

2.2.2 Parametrization of PID controllers

Tuning a PID controller can be a challenging task, depending on the system. The fact that the three parameters have different effects on the typical performance measures can make tuning especially difficult. While increasing the parameter K_P will lead to a decrease in the rise time, it will also result in an increased overshoot. Decreasing K_I results in an increasing overshoot but decreases the settling time. All three parameters must be tuned correctly at the same time to achieve a good performance. As already mentioned, there exist numerous approaches for PID parameter tuning and an extensive overview can be found in [78, 79] and [80]. In this thesis, two classic approaches were mainly used for PID parameter tuning and will briefly be explained in the following paragraphs. Both are offline approaches, meaning that the PID parameters are found first and then applied without changing during the test performance.

Ziegler-Nichols: The Ziegler-Nichols tuning approach [81] was introduced in 1942 and is considered one of the most familiar and easiest tuning approaches in practice [3]. This procedure is based on a heuristic procedure. The K_P parameter is consecutively increased until the system is in consistent and stable oscillation. This state is described by the ultimate gain K_U and the period T_U. Now, the three PID parameters are set to $K_P = 0.6K_U$, $K_I = 0.5T_U$ and $K_D = 0.12T_U$. This parametrization is mainly used for disturbance control, as it creates a quarter wave decay, meaning a reduction of $\frac{1}{4}$ for the amplitude within one oscillation period. It has to be noted that these parameters usually result in an aggressive gain due to the high K_P value, causing a significant overshoot that might lead to instability, which is one of the drawbacks of this tuning approach.

Genetic Algorithms: Genetic algorithms [82] are another example of a heuristic approach that has successfully been applied to PID tuning [83, 84]. There are three main ideas that are combined to form a genetic algorithm: selection, crossover and mutation. Genetic algorithms are supposed to mimic evolutionary processes in order to find a solution. For applying a genetic algorithm, the problem is usually stated in binary. In the case of PID parameter tuning, the PID parameters are converted into binary code. For example, the double standard parameter $K_P = 3.65$ would be translated into the binary 0100000000001101001100110011001100110011001100110011001100110011. Each individual 0 or 1 is called a gene, while the whole sequence is called a chromosome. To start the genetic algorithm, random chromosomes are initialized and represent the whole population. For each member of the population, the fitness is evaluated, according to a fitness function. This function has to be chosen based on the problem at hand. To parametrize a PID controller, the fitness function can be a weighted combination of the typical performance measures, i.e. rise time, overshoot, settling time, steady-state error and stability. A member with higher fitness will be more likely to be chosen for the selection process. The selection is performed, using the roulette wheel approach [85]. The chosen parents are now combined in the crossover: The chromosome of each parents is split at a random point and the two are exchanged, resulting in two offspring that combine the characteristics of both parents. An example would be the two offspring 00011 and 11100 as a result of the crossover of the two parents 00000 and 11111. In a last step, with a very small probability, a random gene is mutated. After the three steps are performed, a new generation is now available for evaluation. This procedure is repeated until a certain performance threshold is met.

Algorithm 1 Pseudo code for Genetic Algorithm

```
initialize: random population at time t
determine: fitness for population at time t
repeat: until best individual is good enough:
    select: parents from population at time t
    perform: crossover on parents, creating population at time t+1
    perform: mutation on population at time t+1
    determine: fitness for population at time t+1
```

Table 2.1: Genetic algorithm for finding PID parameters.

2.3 Machine learning

Machine learning refers to a subfield of computer science that enables machines to learn. For a huge variety of problems, computers can be used in an efficient way to solve them. They are usually empowered to complete these tasks by being programmed. However, some tasks are not easy to be programmed. This might be because they cannot be well defined, because they involve a relationship that is hidden in a huge amount of data, or because they have changing characteristics and unforeseen environmental conditions. Machine learning can help with these sorts of tasks by finding hidden structure and patterns within the data or construct useful approximations for the problem at hand. For example, while sorting numbers can easily be programmed as an algorithm, the task of identifying spam emails is not that easy [86]. Still, the latter task can be formulated in a typical algorithmic formulation: The input is the questionable email, while the output is the classification as spam or non-spam. It is unreasonable to write an algorithm that can identify each possible spam email, but, given the amount of available data, it is better to learn the solution to this task from data. As the machine has the ability to change its inherent model of the problem, this data-driven approach is called machine learning. The machine learning algorithm takes examples for spam and non-spam emails and will create its own rules for handling the problem. These rules might not necessarily be easily understood by humans, depending on the employed machine learning algorithm. Machine learning approaches have recently become more and more applicable to real-world problems due to advances in computer technology. Larger storage capabilities enable us to store more data, and at the same time, that data can be accessed worldwide by networks with increasing bandwidth.

A very wide definition for machine learning is given by [87]: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E." This definition already indicates that an individual machine learning algorithm only performs well for an individual task, which is known as the *no free lunch theorem* [88]. Each individual task requires its own machine learning algorithm with a period of learning from samples in order to perform well on unknown samples from the same problem. Therefore, to solve complex problems in a semi-intelligent way, it might be necessary to combine different machine learning algorithms.

Machine learning approaches have in common that they do not rely on a human provided model to learn and perform. These techniques are purely data-driven and can therefore be applied to every problem that provides a sufficient amount of training data. Machine learning algorithms prove to be especially successful for tasks that humans consider mentally difficult but can be described by mathematical rules, like chess [89] or Go [90]. It is only recently that more intuitive problems like face and image recognition have successfully been solved by machine learning algorithms with human-like performance or even with better performance [91, 92].

To be useful, machine learning algorithms have to be able to extract important information from the provided data and learn to generalize from them. This means that the algorithm should perform well on new, previously unseen data. In a typical setting, the data is provided as a data set. This data set is used to train the machine learning algorithm and is therefore called training set. The training itself is performed by adapting the parameters of the algorithm to minimize the error between the training set and the algorithms output. The result is a performance measure, called the training error. This measure, however, does not give information about the ability to generalize. To evaluate the algorithm's generalization capacity, it needs to perform well on unseen samples. Therefore, new data from a so called test set is used to calculate another performance measure, the test set error. Eventually, the algorithm's training and test set error should be small. If the algorithm's training error remains big, the algorithm is underfitting the data. Its capacity is not big enough to contain all relevant information to learn the problem. If the training error is small, but the test error is big, the algorithm is overfitting. It is too specialized on the training data and is no longer able to generalize [93].

Splitting the limited amount of available data unfortunately leads to another problem. As machine learning algorithms rely on data to learn their models, they perform better the more data they can sample. The more data is moved from the training set to the test set, the worse the algorithms performance will become. In addition, this approach is highly susceptible to variance as the training and test error strongly depend on the data, which is assigned to the test set and the training set. In the worst case, an important phenomena might only be included, either in the training set or in the test set, resulting in a poor performance. To avoid this dilemma, cross validation is applied. Here, the data is divided into training and test samples repeatedly and the average over all training-test splits is calculated. Figure 2.5 demonstrates a leave one out cross validation for a data set with K samples. In each run, only one sample is moved into the test set and the remaining samples form the training set. Therefore, each sample in the whole data set will be tested during the cross validation. This approach makes the best use of the data. After the best

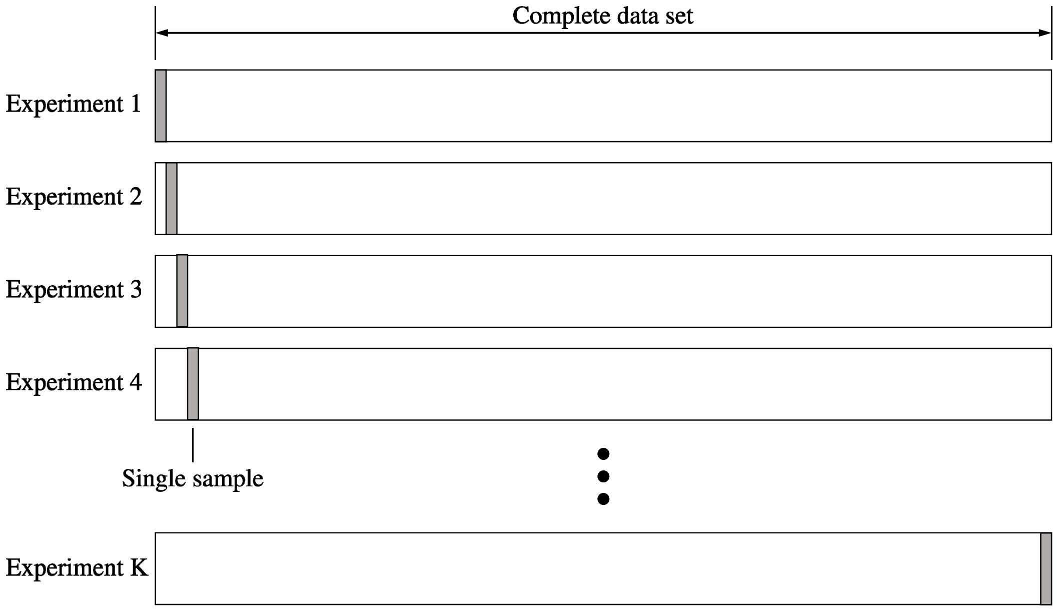


Figure 2.5: Leave one out cross validation procedure.

model is evaluated by using cross validation, the model can be learned on all available data and then be applied to new data.

Machine learning has close relations to some fields of traditional math and engineering. Numerous methods of machine learning use statistical approaches and measures for analyzing data. Another source of inspiration for machine learning are biological systems. The functioning of neural networks is closely related and inspired by neurons in the human brain [94], while the mechanism of reinforcement learning have been shown to resemble the function of dopamine [95]. Lastly, a natural application for machine learning is adaptive control theory, as control processes inherent a certain degree of uncertainty and are prone to changing conditions and dynamics.

Machine learning algorithms can be classified into two major classes, called unsupervised learning and supervised learning. The distinction is made, based on the provided feedback signal. In supervised learning the data is traditionally available as labeled data and expert knowledge is available. In unsupervised learning the data is usually not labeled.

2.3.1 Supervised learning

Supervised learning is defined by a machine learning class, where a feedback signal is provided, and is primarily a case of learning from labeled examples. The general task for a supervised algorithm is to find a mapping, following the equation

$$y = f(x, w), \tag{2.3}$$

where x is the input, w are the model parameters and y is the output. One of the most common problems for supervised learning is the classification task [96, 97, 98]. Following the notation of Equation 2.3, x would be the data to be assigned to a class and y the class to be assigned to. For the algorithm to learn the mapping, i.e. adjusting the parameters, it has to be provided with a labeled data set. This set consists of labeled samples, where the true class is known for each input sample. Eventually, the learned mapping function $f(x, w)$ should be usable as a reliable classifier for new input samples.

To evaluate the performance of the algorithm as a classifier a second labeled data set, the test set, is used. The inputs of this second data set are fed into the classifier and the classification rate, i.e. the percentage of correct classified samples, is measured. This gives information about how well the algorithm learned the classification task.

Supervised machine learning algorithms are not limited to classification tasks, however. Another application for this type of algorithm are prediction tasks, which are also named regression tasks. For this type of task the algorithm does not try to find the correct class for a given sample, but rather a correct associated numerical value. The training and test procedure still remain the same.

Supervised learning algorithms include, but are not limited to, logistic regression approaches [99], support vector machines [100], neural networks [101] and non-parametric algorithms, like k-nearest neighbor [102] or decision trees [103]. A detailed overview over supervised machine learning techniques can be found in [104, 105].

2.3.2 Unsupervised learning

Unsupervised learning refers to a class of machine learning algorithms that learn without a feedback or supervision signal. As an unsupervised machine learning algorithm still needs a measure to evaluate its performance, these type of algorithm usually try to maximize or minimize a value that is contained in the data set itself. This performance measure strongly depends on the specific application.

A typical task for an unsupervised machine learning algorithm is to find a better representation for data. The term “better” is hereby strongly dependent on the application. Possible better representations include low-dimensional representations, sparse representations or invariant representations or a combination of these attributes. There exist a huge variety of algorithms to find new representations. It includes simple statistic approaches, like the PCA [106], sparse approaches [107] or autoencoding neural networks [108, 109].

Another common task for unsupervised learning is clustering. When using a non-hierarchical approach like k-means [110] clustering, the data is partitioned into k independent clusters. The approach of clustering can be extended to not only find clusters within the structure of the data, but also generate a hierarchical structure [111]. This way, the data set can be better understood and visualized as data points that are similar will be assigned to the same cluster. Clustering can furthermore be used for anomaly detection. Under the assumption that the majority of the data is normal and the anomalies are qualitative different from the normal data, it can easily be checked whether a new sample is an outlier or not.

2.4 Neural networks

The concept of artificial neural networks was introduced in 1958 by Frank Rosenblatt with his perceptron [94]. A single perceptron can be viewed as the smallest version of an artificial neural network. It is limited to linear computations. The output of a perceptron can be calculated as the solution to the equation

$$y = f\left(w^{\top} x + b\right), \tag{2.4}$$

where $x \in \mathbb{R}^p$ are the input signals, $w \in \mathbb{R}^p$ are the corresponding weights, $b \in \mathbb{R}$ is a bias and f is an activation function. If several perceptrons are connected to each other and ordered in different layers, every differentiable function can be approximated. This structure is called multilayer perceptron (MLP) and can be seen in Figure 2.6. The multilayer perceptron is a simple feed-forward neural network that consists of multiple layers of perceptrons, that are visited once from the input to the output. In the given example, they are ordered into three layers. The first layer acts as the input layer, the second layer is a hidden layer, mapping the inputs based on the connection weights and the bias, while the third layer acts as an output layer. The MLP is capable of representing more complex functions than a simple perceptron. The most simple form of neural networks are strict feed-forward networks. Here, each neuron is only connected to the neurons in the previous and subsequent layer. From equation 2.4 can be seen that the output for a given input is only dependent on the weights w and biases b. These are called parameters and have to be learned. The standard learning algorithm for a feed-forward neural network is backpropagation [112]. In backpropagation, the input is fed into the network and processed during a forward pass. During the forward pass, each neuron has a certain activity, based on its corresponding weights and the chosen activation function. At the output layer, the actual output is compared to the desired output and the error is calculated as the solution of a loss function. Such a loss function is usually chosen based on the learning problem.

An example of a loss function is the quadratic loss function given by the equation

$$L(y, y^*) = \frac{1}{2}\sum_{i=1}^{n}(y_i^* - y_i)^2, \tag{2.5}$$

where y^* is the desired network output, y is the network output and n is the number of samples. The error is then propagated backwards through the network, according to the activation of each neuron in the forward pass and the weights are adjusted, based on a gradient descent method. To further improve the learning of neural networks, various techniques have been invented. The training samples are usually provided in form of batches [113] to speed up the training process and improve convergence. Further improvement can be achieved by applying a momentum [114, 115]. In order to achieve a better generalization and prevent overfitting, for fully connected neural networks a regularization term in the form of a weight decay is added [116, 117]. For real valued inputs, such as images, the suggested activation function is the linear rectifier function [118, 98, 119], as it does not clinch the input, leads to a sparser representation [120] and improves the backpropagation performance.

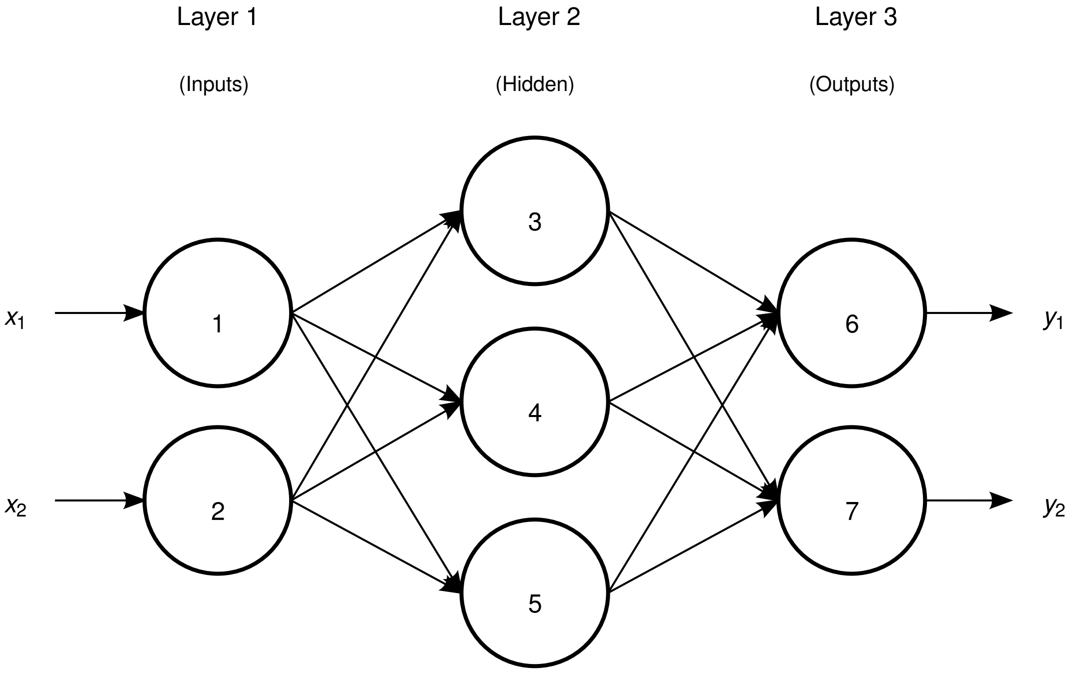


Figure 2.6: A multilayer perceptron with 7 neurons, arranged in 3 layers.

2.4.1 Autoencoder

An autoencoder is a neural network, arranged in a special way. The most simple autoencoder consists of three layers: the input layer, one hidden layer and an output layer. The number of neurons in the hidden layer is smaller than the number of neurons in the input layer, forcing the information of the input to be compressed. The output layer will have the same amount of neurons as the input layer. As input and output are of the same dimension, they can be compared for an unsupervised learning approach. If input and output are identical, all information has been kept during the transformation and compression in the smaller hidden layer, yielding a compact representation. Given a number of p neurons in the input and output layer and q neurons in the hidden layer, the feature value at the i-th neuron in the hidden layer can be calculated as solution of the equation

$$h_i(x) := f\left(w_i^\top x + b_i\right), \qquad \text{for all } i = 1, \dots, q, \tag{2.6}$$

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

$$h := [h_1(x), h_2(x), \dots, h_q(x)]^\top \in \mathbb{R}^q \tag{2.7}$$

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

$$y_j(h) := f'\left(w_j^{'\top} h + b_j'\right), \qquad \text{for all } j = 1, \dots, p, \tag{2.8}$$

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 $W := [w_1, \dots, w_q] \in \mathbb{R}^{p\times q}$ and $W' := [w_1', \dots, w_p] \in \mathbb{R}^{q\times p}$. It has to be noted 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 [108]. The loss function for an autoencoder is slightly different to the loss function of a regular feed-forward neural network, as encoder and decoder do not necessarily share the weights, and follows the equation

$$L(x, y, w, w') = \frac{1}{2}\sum_{i=1}^{n}(x_i - y_i)^2 + \frac{\lambda_1}{2}\|w\|_F^2 + \frac{\lambda_2}{2}\|w'\|_F^2, \tag{2.9}$$

where x is the input, y is the output, W and W' are the weights, λ is a regularizing term and $\|\cdot\|_F$ is the Frobenius norm of matrices. This loss function minimizes the difference between the input and the output, that has been computed based on the hidden representation. It can therefore be seen as a measure for the quality of the hidden representation.

A more extensive introduction on fully connected denoising stacked autoencoders can be found in [108]. Similar to the development in neural networks for classification, the technique of convolutional neural networks have been extended to autoencoding as well [121, 122].

In this thesis, stacked denoising autoencoders and stacked convolutional autoencoders will be used and explained in more detail in Section 4.3.

2.4.2 Convolutional neural networks

Convolutional neural networks were firstly applied to computer vision problems. This type of network performs especially well on data that has a known grid-like structure [93], e.g. time-series data or images. They are inspired by the receptive fields within the visual systems of cats [123, 124]. Similar to the visual cortex in these animals, neurons are not connected to all neurons in the previous and subsequent layers (fully connected), but only to a small subset of neurons [125]. The two basic ideas for changing the connection between the neurons in convolutional neural networks are sparse connectivity and weight sharing and can can be seen in Figure 2.7. On the left side, the sparse connectivity—only three neurons are connected to a neuron in a subsequent layer— is depicted. This leads to a dramatic decrease in parameters, speeding up the learning process and eliminating numerous local minima in the optimization process. The right side shows the shared weights, indicated by connections of the same color. The idea of weight sharing leads to the same parametrization in terms of weight *w* and bias *b* for the similar connections between the layers. Together, these ideas generate a feature map that identifies patterns throughout the complete input. Mathematically, this can be expressed as a convolution, giving the name for this type of neural network. The convolutional layers are the first essential building blocks for convolutional neural networks. While the filters are usually randomly initialized they are learned autonomously during the training process. When trained on patches of natural images, these filters tend to become gabor-like [126, 127, 128]. The influence of different filters on the convolved image can be seen in Table 2.2. The second essential building block for convolutional neural networks is max pooling, which is depicted in Figure 2.8. Within a given size, only the pixel with the highest value is passed further through the neural network. Max pooling has two essential functions: First, it creates an

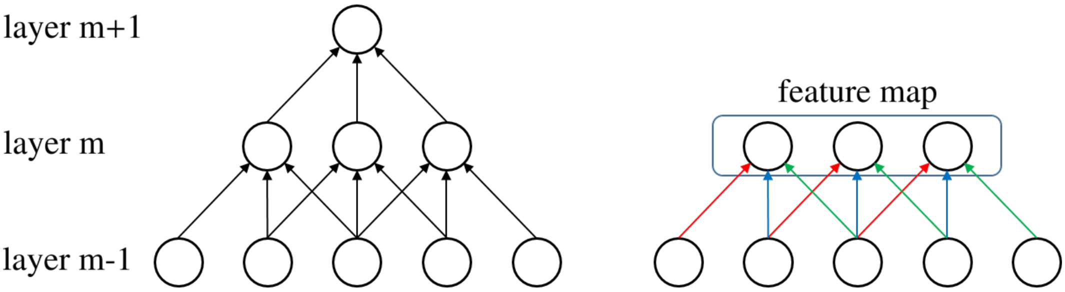
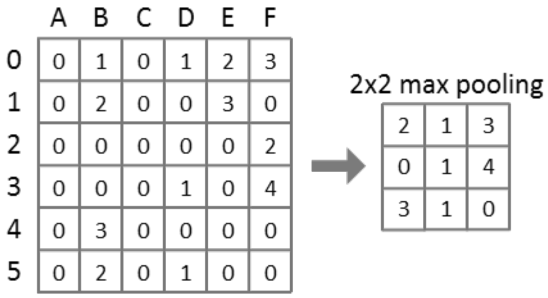


Figure 2.7: Concept of sparse connectivity (left) and weight sharing (right).

invariance to translation, rotation or scaling. Within the range of the max pooling mask, the pixel with the highest value can change its position, without changing the reaction of the neural network. It can therefore experience small amounts of translation, rotation or scaling without any effect for the later representation. Second, by applying max pooling, the representation of the input is compressed, dependent on the size of the max pooling mask. This significantly decreases the computational complexity and in addition prevents the neural network from overfitting. The first convolutional neural network that caught significant attention from the computer vision community was the “AlexNet” [98]. Since then, convolutional neural networks have won all important computer vision challenges and even demonstrated human like performance or outperformed them for specific tasks [91, 129]. Convolutional architectures can be used in a supervised way for image classification as well as in an unsupervised way for autoencoders [130].



	A	B	C	D	E	F
0	0	1	0	1	2	3
1	0	2	0	0	3	0
2	0	0	0	0	0	2
3	0	0	0	1	0	4
4	0	3	0	0	0	0
5	0	2	0	1	0	0

2	1	3
0	1	4
3	1	0

Figure 2.8: 2 × 2 max pooling with a stride of 2.

2.4.3 General dynamic neural networks

Regular feed-forward networks are defined by the property that no neuron can be visited twice from the input to the output. By this definition, neural networks with shortcuts and lateral connections [131] still remain feed-forward networks. Simple feed-forward networks have proven to learn a sufficient model for most Markov problems. For non-Markov problems however, taking time into consideration significantly improves the performance of neural networks. This led to the first neural networks that were not feed-forward networks. By feeding the output back into the input layer via a so called ‘context neuron’ a superior performance was achieved. Additionally the context neurons in the input layer are recurrently connected to themselves. By introducing recurrent connections within the neural network, the output is no longer only dependent on the input but also on previous inputs. The neural network has gained a short-time memory. This type of network has been named Jordan networks [132]. To avoid an algebraic loop and make this network computable, the context neurons have delay of 1. Jordan networks are applied to different problems and proven to be superior to simple feed-forward neural networks for some tasks [133, 134, 135]. A slightly different configuration that was introduced to specifically deal

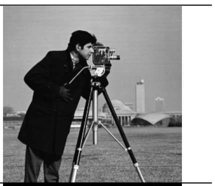
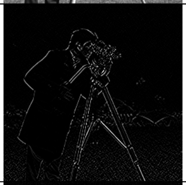
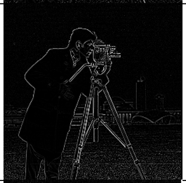
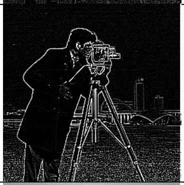
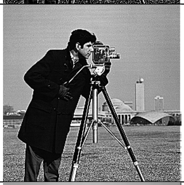
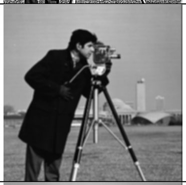
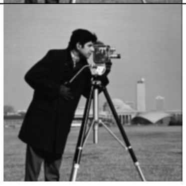
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur	$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$	
Gaussian Blue	$\begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix}$	

Table 2.2: Convolutions with different filter, image courtesy of Massachusetts Institute of Technology.

with temporal structure are Elman networks that were presented in 1990 [136]. In this type of network the context neurons do not receive their input from the output layer but from the first hidden layer. Elman networks demonstrated to successfully make use of temporal structure for different applications [137, 138]. So far, the neural networks had a very specific architecture and are special cases of general dynamic neural networks. Here, it is possible to connect each neuron with each neurons with arbitrary delays. This degree of freedom introduces a wide range of neural network architecture, even if the neural network consists only of a small number of neurons. An example of a general dynamic neural network is given in Figure 2.9. For a general dynamic neural network, a two dimensional adjacency matrix, i.e. the matrix containing all weights, does not suffice. For each new time delay, a new two dimensional adjacency matrix is needed, creating a three dimensional matrix to represent the weights for such a neural network. A three dimensional adjacency matrix can be seen in Figure 2.10. The corresponding neural network consists of n neurons and ψ delays, represented by $n \times n \times \psi$ matrix.

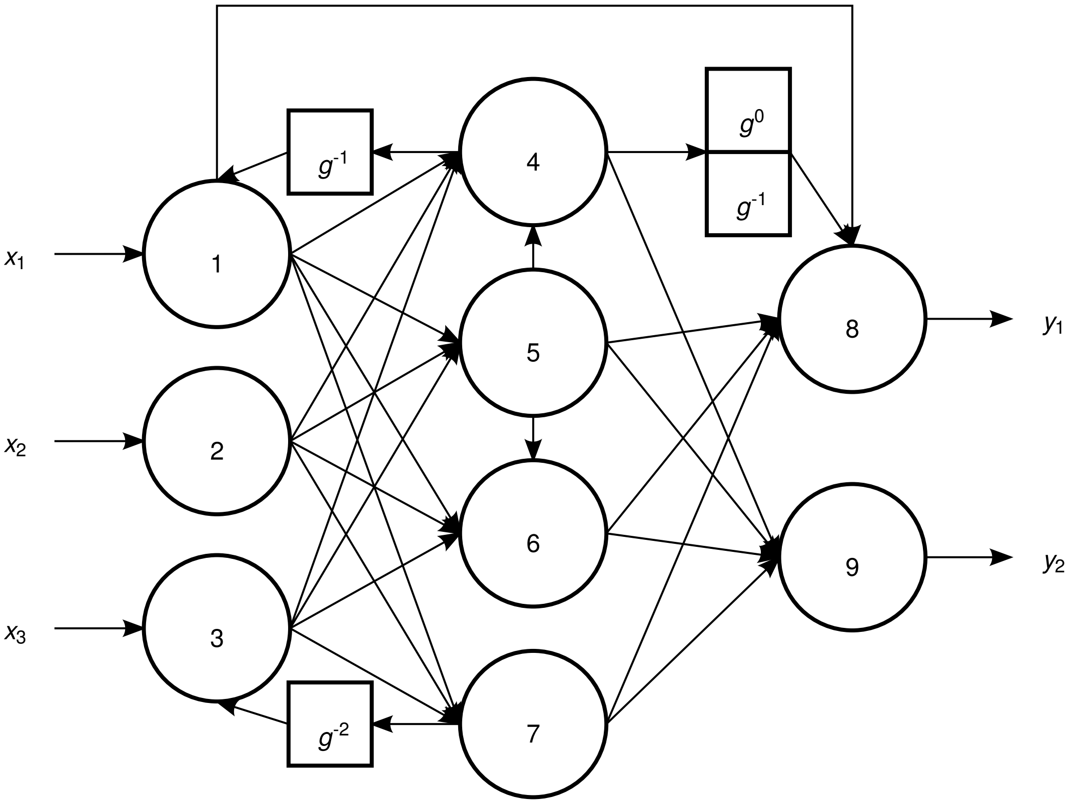


Figure 2.9: A general dynamic neural network, including shortcuts and recurrent connections.

To apply backpropagation to a neural network that is not a feed-forward neural network, it has to be temporally unfolded [139]. The unfolding of a recurrent neural network can be seen in Figure 2.11. Temporally unfolding means that the neural network is transformed into a feed-forward neural network by replicating it. A neural network with a recurrent con-

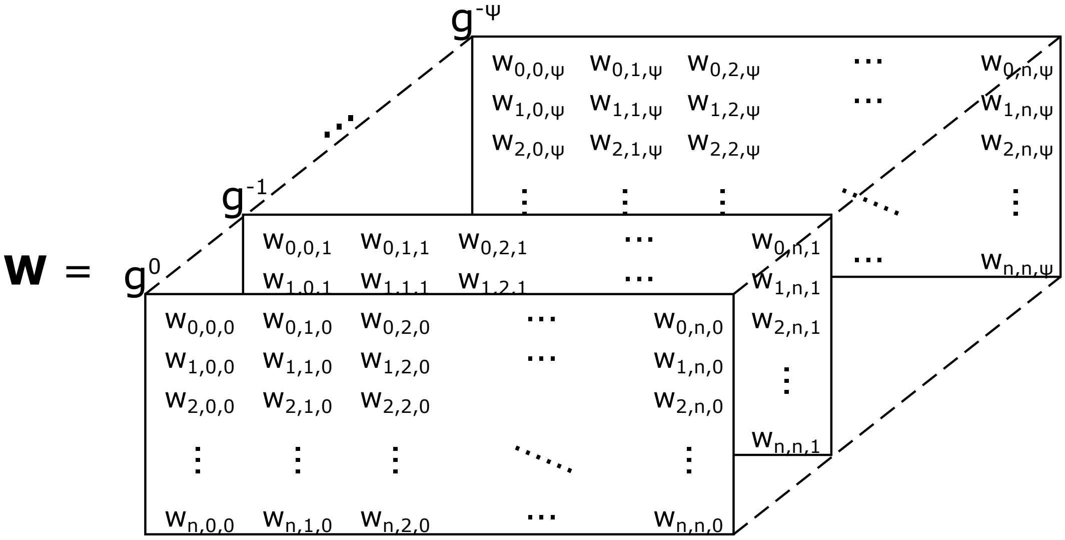

W =
$g^{-\psi}$
g^{-1}
g^{0}
$w_{0,0,\psi}$ $w_{0,1,\psi}$ $w_{0,2,\psi}$ ⋯ $w_{0,n,\psi}$
$w_{1,0,\psi}$ $w_{1,1,\psi}$ $w_{1,2,\psi}$ ⋯ $w_{1,n,\psi}$
$w_{2,0,\psi}$ $w_{2,1,\psi}$ $w_{2,2,\psi}$ $w_{2,n,\psi}$
⋯ $w_{n,n,\psi}$
$w_{0,0,1}$ $w_{0,1,1}$ $w_{0,2,1}$ ⋯ $w_{0,n,1}$
$w_{1,0,1}$ $w_{1,1,1}$ $w_{1,2,1}$ ⋯ $w_{1,n,1}$
$w_{2,n,1}$
$w_{n,n,1}$
$w_{0,0,0}$ $w_{0,1,0}$ $w_{0,2,0}$ ⋯ $w_{0,n,0}$
$w_{1,0,0}$ $w_{1,1,0}$ $w_{1,2,0}$ ⋯ $w_{1,n,0}$
$w_{2,0,0}$ $w_{2,1,0}$ $w_{2,2,0}$ $w_{2,n,0}$
$w_{n,0,0}$ $w_{n,1,0}$ $w_{n,2,0}$ ⋯ $w_{n,n,0}$

Figure 2.10: A three dimensional adjacency matrix for a neural network with n neurons and ψ delays.

nection has to be replicated g times for $g+1$ time steps. Special cases of general dynamic neural networks, i.e. recurrent neural networks, have shown their capabilities for solving nonlinear convex programming problems [140] or handwriting recognition [141].

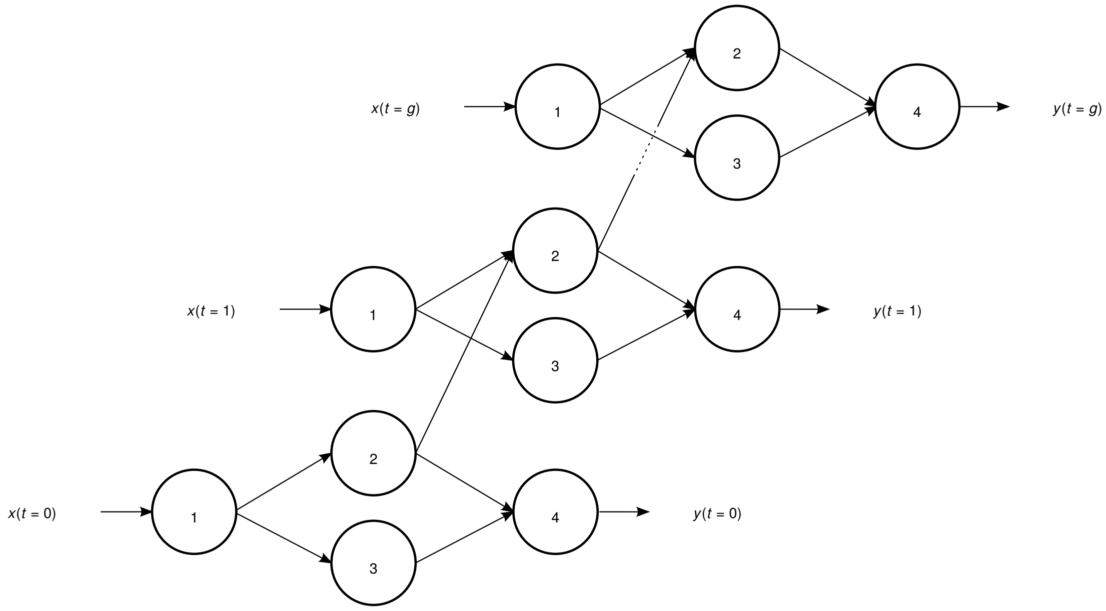


Figure 2.11: A neural network with a delay of one is temporally unfolded for $g + 1$ time steps.

2.5 Reinforcement learning

Reinforcement learning (RL) is a branch of machine learning and artificial intelligence that is not characterized by specific learning methods, but rather driven by the problem to be learned. It focuses on goal-directed learning and decision making. While the optimization function in other machine learning approaches is often specifically defined for the algorithm at hand, reinforcement learning is always concerned with the long time maximization of one particular signal—the reward signal. This scalar signal provides the reinforcement learner with an indicator for success or failure. It does not need to be provided by an external supervisor, thus enabling the reinforcement learner to learn on its own by trial and error.

Interactions with the environment happen on a simple basis of sensory sensation, taken action and collected reward, which can be seen as the most intuitive and simplest form of learning. This simple setup is depicted in Figure 2.12. The reinforcement learning setup is similar to a regular closed loop control approach, except for the set point and the error being delivered by the same signal—the reward. As the reinforcement learning agent has the capability of evaluating its own decisions, i.e. the actions taken, it can also change the way these actions are selected to improve its performance. By assigning values to specific sensory sensations and also the sensation-action pair, the reinforcement learning agent can compare those values and improve its action-selection. This again will change the assigned values for the sensations, resulting in an iterative process.

The mapping from sensations (states) to actions is called *policy* and the assigned evaluation of a states value is called *value function*. An important attribute of reinforcement learning is the fact that not only immediate rewards are taken into account but also the long time performance. The reinforcement learning agent might at some point decide to take an action with a smaller immediate reward to transit into a more promising state. This long term planing turns out to be a significant feature for problems that require complex solutions rather than step by step optimization. Together with the ability of learning from ongoing sequences of experience, reinforcement learning is well suited for changing environments and varying conditions. There are two major developments in reinforcement learning: The first is the usage of general value functions to learn and apply predictions from samples of experience (prediction learning), while the second uses samples of experience to adapt a policy in order to maximize the outcome (control learning). Examples of reinforcement learning include problems such as the mountain car problem [142], robotics [143, 144] or playing computer games [145].

To formalize the reinforcement learning problem, the following subsections will introduce the underlying assumptions and introduce the most important concepts of reinforcement learning that have been utilized in this thesis.

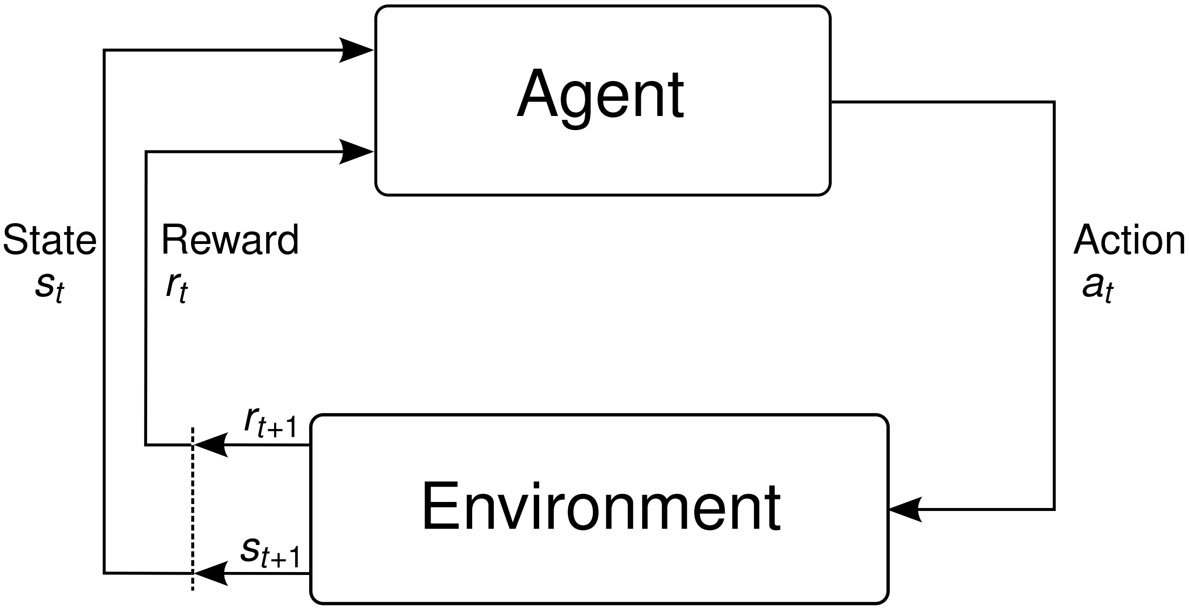


Figure 2.12: The general reinforcement learning framework [146].

2.5.1 Markov decision processes

Reinforcement learning assumes each problem at hand to resemble a Markov decision process. The learner progresses through states by taking actions and receives rewards during the transition. This can be described by the quintuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. Here, $\mathcal{S}$ denotes 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, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ the rewards observed on these transitions and $\gamma \in [0, 1]$ the discount rate. At each time step $t \in \{0, ..., T\}$, where T is the final time step, the learner chooses an action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$, following a policy $\pi : \mathcal{S} \to \mathcal{A}$. This action will result in a successor state $s' \in \mathcal{S}$ and a reward $r = \mathcal{R}(s, a, s') \in \mathbb{R}$, according to the transition probabilities and rewards given by $\mathcal{P}$ and $\mathcal{R}$. Within this environment, the reinforcement learning agent tries to maximize the expected sum of future rewards, which is called return and defined as

$$g_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{2.10}$$

where γ is the discount rate that weights immediate reward against future reward. If $\gamma < 1$, then the sum of future rewards is bounded. It is a measure for how far-sighted the algorithms is. By adapting γ, the learner can range from considering only immediate rewards ($\gamma = 0$) to taking the complete future into account ($\gamma = 1$). Reinforcement learning divides problems into episodic and continuous tasks. While an episodic task has a natural ending that occurs as soon as a final state is reached, continuous tasks do not have a natural ending. If $\gamma = 1$ in a continuous tasks, the expected return is not bounded, but approaches infinity. While the agent samples the environment, according to a policy π, it learns a value function $v_\pi : \mathcal{S} \rightarrow \mathbb{E}_\pi[g_t \mid s_t = s]$ that is a mapping from states to the expected return or a state-action value function $q_\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{E}_\pi[g_t \mid s_t = s, a_t = a]$ that maps from states and actions to the expected return. These functions are approximations of the true expected return and are used to subsequently improve the policy π. As value function and policy depend on each other and improve each other, this scheme is an iterative process, called general policy iteration. Eventually, the policy π and the value function v_π or q_π are the outputs or solutions of reinforcement learning.

2.5.2 Value functions and general value functions

A value function describes an estimate of the discounted sum of future rewards. It can be defined as

$$v_\pi(s) = \mathbb{E}_\pi[g_t \mid s_t = s] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s], \tag{2.11}$$

where $\mathbb{E}_\pi[g_t \mid s_t = s]$ denotes the expected return, following policy π and starting in state s. This relationship can also be described by the Bellman equation in a recursive scheme, following equation

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \sum_{s' \in \mathcal{S}} \mathcal{P}_\pi(s, a, s')[r(s, a, s') + \gamma v_\pi(s')], \tag{2.12}$$

where $\mathcal{P}_\pi(s, a, s')$ is the probability of transitioning from s to s' while taking action a and $v_\pi(s')$ is the value of the successor state s'. Value functions are closely related to decision making in the traditional reinforcement learning setting. The value function is a means to determine a policy π for a given problem in order to maximize the expected long term return $\mathbb{E}_\pi[g_t \mid s_t = s]$. Once the value function has converged towards the true values, a greedy policy is sufficient for decision making, as an optimal value function with a greedy policy will result in optimal control.

General value functions on the other hand are not necessarily related to control questions, but to predictions. Their purpose is to learn dependencies between states and predictions about the environment. The setting for a general value function is essentially the same—the agent experiences a quadruple of (s, a, r, s'), while sampling the process. Here, r is the signal of interest to be learned. It is called pseudo-reward or cumulant. [147]. The general value function is a mapping from state s to prediction v, with the three auxiliary inputs policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, discount rate $\gamma \in [0, 1]$ and the cumulant $r : \mathcal{S} \rightarrow \mathbb{R}$ and is defined as

$$v(s, \pi, \gamma, r) = \mathbb{E}_\pi[g_t \mid s_t = s], \tag{2.13}$$

where g_t is the return, calculated from the discounted cumulants.

The performance of the learned general value function is the difference between target (discounted cumulants) and prediction. As the cumulants can be arbitrary signals of interest, the general value function can learn and represent potentially useful predictive knowledge about the environment [147]. Furthermore, by applying classical reinforcement learning, we can benefit from its scalability and performance guarantees.

As the return is not only defined by the future sum of rewards, but also a discount factor γ, so is the target in the general value function setting. Although a changing γ results in a broader range of potentially predictive knowledge, in this thesis only a fixed γ will be used. Such a fixed γ will result in a prediction with a fix time horizon, allowing a predictive control approach. In order to learn an approximate general value function, tile coding and classical temporal-difference learning can be used. An important advantage of general value functions is the fact, that due to function approximation the learned value function does neither need extensive computation, nor memory. It is therefore possible to learn multiple general value functions on different time scales simultaneously. A comprehensive introduction and further applications for general value functions and predictive knowledge in reinforcement learning can be found in [147].

2.5.3 Linear function approximation

Almost all machine learning approaches suffer from the curse of dimensionality, when applied to real-world problems [148] and so does reinforcement learning [142]. Even a relative simple game like chess has an estimated number of 10^{56} possible positions that can be reached [149]. It is therefore not only reasonable but necessary to introduce a way of approximating states in order to make problems computationally solvable. Although there exist a variety of potential function approximation techniques, linear function approximation is one of the most commonly used approaches in reinforcement learning [150, 151]. One drawback of linear function approximation is the limited capability to represent nonlinear dependencies. This can be compensated by using tile coding, which includes nonlinear basis functions. The eventual approximate value function will still be linear, but the representation of nonlinear problems can be improved significantly. A typical approach is to linear approximate the learned value function, according to the equation

$$v(s) \approx \hat{v}(s, w) = w^\top \chi(s) = \sum_{i=1}^{n} w_i \chi_i(s), \tag{2.14}$$

where $w \in \mathbb{R}^n$ is a vector of weights and $\chi(s) \in \mathbb{B}^n$ is a binary, sparse tile coded state vector and n is the memory size. As this function approximation is linear with respect to the weights w, it is guaranteed to converge to or near a global minimum, when used in combination with gradient descent. The gradient can simple be calculated as the solution of the equation

$$\nabla_w \hat{v}(s, w) = \chi(s), \tag{2.15}$$

where $\nabla_w \hat{v}(s, w)$ is the derivative with respect to the weights. Linear function approximation can also be used to approximate action-value functions or policies in a policy gradient setting.

2.5.4 Policy gradient

Traditionally, reinforcement learning algorithms were used to learn an (approximated) value function, which could then be utilized to derive a policy. These value functions might have been either state value functions or state-action value functions. Instead of taking a "detour" via the value function, policy gradient approaches directly learn a parametrized policy $\pi(a \mid s, \theta)$, depending on the state and the learned weights. Here θ are the weights for the policy. During the learning the weights are adapted, based on the gradient of a performance measure $\eta(\theta)$ with respect to the policy weights to maximize the algorithms performance, i.e. the expected return as solution of the equation

$$\theta_{t+1} = \theta_t + \alpha \nabla_\eta(\theta_t), \tag{2.16}$$

where $\nabla_\eta(\theta_t)$ is a stochastic estimate. However, policy gradient approaches do not exclude learning an additional value function, as can be seen in actor-critic algorithms that combine learning a parametrized policy with learning a value function. Policy gradient has several advantages, compared to value function learning algorithms. Most value function learner use a greedy or an ϵ-greedy action selection. A greedy action with respect to the value function is the action that maximizes the return. An ϵ-greedy approach chooses a random action with the probability of ϵ and the greedy action otherwise. To still ensure exploration, usually ϵ is set to be bigger than zero. Policy gradient algorithms can use any representation for the policy (including neural networks or statistical distributions) that is differentiable with respect to the weights. This allows a much broader selection. Additionally, these policies can eventually become deterministic, which is not the case for ϵ-greedy policies. For some problems, a policy might be significantly easier to learn than a value function, as problems significantly vary in their complexity of policies and action-value functions. One example of this problem might be a gridworld, where a simple policy performs better than a state-value function. Furthermore, state-action value functions cannot find stochastic policies, which can easily be achieved in a policy gradient algorithm. Finally, by choosing an appropriate initialization for the weights, prior knowledge about potentially good policies can be included.

One particular policy gradient algorithm that will be used in this thesis is an actor-critic algorithm. It learns a parametrized policy and at the same time a state value function. Using the regular (bootstrapping) temporal-difference learning makes it especially well suited for learning from incomplete samples and online learning. As actor-critic algorithms can be combined with statistical action selection, they are well suited for huge and continuous actions spaces. In this thesis, an actor-critic algorithm will be used for learning the choose the correct laser power for a laser welding problem. The algorithm will be presented in more detail in Section 4.5, including the stochastic action selection approach and the details for the specific problem.

3 General dynamic neural networks for stochastic adaptive control

As already presented, the standard PID controller, although reliable, relatively easy to implement and to understand, has certain limitations that render its application limited. To extend its capabilities beyond the scope of linear and linearized systems, it can be extended by machine learning. To keep the control loop as simple as possible and only rely on the closed loop provided data, the machine learning algorithm is integrated in a way that it receives the same information as the PID controller. The outputs of the machine learning algorithm are the suggested PID parameter, K_P, K_I and K_D for the corresponding machine learning input, namely the control error and the systems output. Therefore, no new sensors are required and the extension only requires access to the PID parameters. This approach is purely data-driven and does not rely on additional models or filters in contrast to adaptive control algorithms. The suggested approach can be seen in Fig. 3.1. The algorithm then adjusts the PID controllers parameter, while the plant is running in order to improve the control performance. The input for the machine learning algorithm is the systems output $z \in \mathbb{R}^k$, where k is the number of outputs and the control error $e \in \mathbb{R}^k$.

3.1 PID tuning by general dynamic neural networks

The implementation of a machine learning based PID tuner will use a general dynamic neural network. For a plant with k outputs, k neural networks will be implemented within the control loop. Each applied neural networks will have 2 input neurons, one for the control error $e \in \mathbb{R}^k$ and one for the systems output $z \in \mathbb{R}^k$. Each neural network will also have 3 output neurons, one for each PID parameter, resulting in $3k$ output parameter in total.

For the setup shown in Figure 3.1, backpropagation cannot be applied naturally, even if the neural network would be temporally unfolded. For applying backpropagation, the error of the neural networks output would be required to be propagated backwards through the neural network. To calculate this error the ideal parameters for the PID controller would be required for all possible states. Another approach is to use the error of the system to be controlled, which only requires the systems output z and the set point z^* and is in fact the control error e. To apply a backwards pass through the neural network, the partial derivative of the error with respect to the weights would be required. As the system cannot be assumed to be differential or even known, this approach turns out to be not practicable. Therefore, the Jacobian matrix of the general dynamic neural network with p

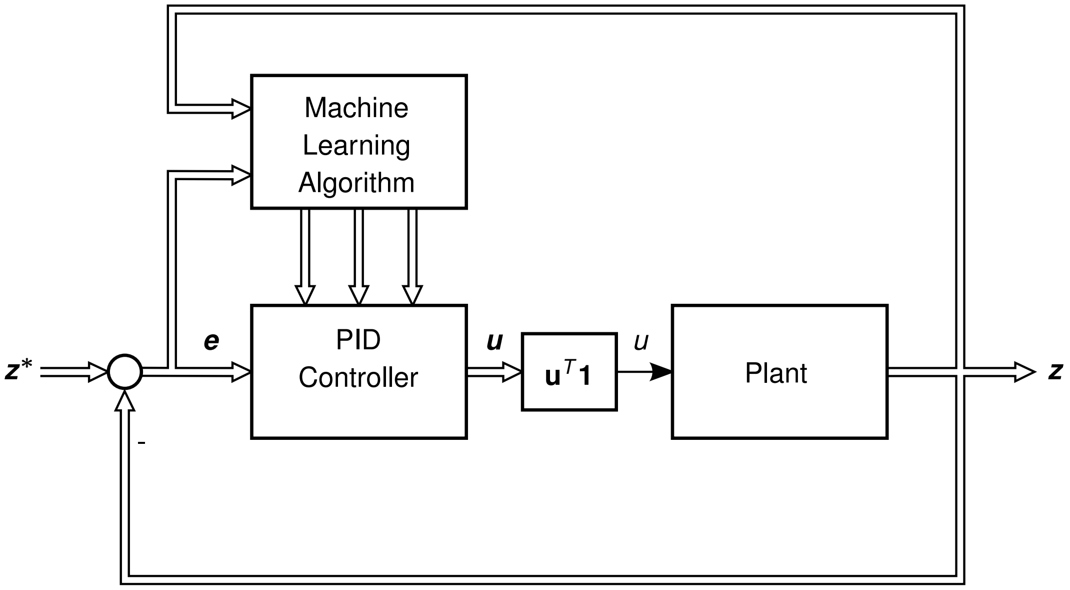


Figure 3.1: The PID auto-tuning structure, including a machine learning algorithm. A machine learning algorithm is added into the control loop, using the same information as a standard PID controller.

input neurons, q output neurons, g time steps and ν weighted connections was calculated numerically according to the equation

$$J_w\big(z(x,w)\big) = \begin{bmatrix} \frac{\partial}{\partial w_0} z_0(x_0,w) & \frac{\partial}{\partial w_1} z_0(x_0,w) & \cdots & \frac{\partial}{\partial w_\nu} z_0(x_0,w) \\ \frac{\partial}{\partial w_0} z_0(x_1,w) & \frac{\partial}{\partial w_1} z_0(x_1,w) & \cdots & \frac{\partial}{\partial w_\nu} z_0(x_1,w) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial w_0} z_0(x_p,w) & \frac{\partial}{\partial w_1} z_0(x_p,w) & \cdots & \frac{\partial}{\partial w_\nu} z_0(x_p,w) \\ \frac{\partial}{\partial w_0} z_1(x_0,w) & \frac{\partial}{\partial w_1} z_1(x_0,w) & \cdots & \frac{\partial}{\partial w_\nu} z_1(x_0,w) \\ \frac{\partial}{\partial w_0} z_1(x_1,w) & \frac{\partial}{\partial w_1} z_1(x_1,w) & \cdots & \frac{\partial}{\partial w_\nu} z_1(x_1,w) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial w_0} z_1(x_p,w) & \frac{\partial}{\partial w_1} z_1(x_p,w) & \cdots & \frac{\partial}{\partial w_\nu} z_1(x_p,w) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial w_0} z_q(x_0,w) & \frac{\partial}{\partial w_1} z_q(x_0,w) & \cdots & \frac{\partial}{\partial w_\nu} z_q(x_0,w) \\ \frac{\partial}{\partial w_0} z_q(x_1,w) & \frac{\partial}{\partial w_1} z_q(x_1,w) & \cdots & \frac{\partial}{\partial w_\nu} z_q(x_1,w) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial w_0} z_q(x_p,w) & \frac{\partial}{\partial w_1} z_q(x_p,w) & \cdots & \frac{\partial}{\partial w_\nu} z_q(x_p,w) \end{bmatrix} \in \mathbb{R}^{gq\times\nu}, \qquad (3.1)$$

where $x_i \in \mathbb{R}^p$ is the input vector at time step i, $w \in \mathbb{R}^\nu$ is the vector of weights which describes the network topology and z_i is the i-th output of $z(x,w) \in \mathbb{R}^q$. According to

Algorithm 2 Compute Jacobian numerically

```
Input: Dynamical system $z(x, W)$, inputs $x$, weights $W$
Output: Estimate of Jacobian matrix $\hat{J}_W$
foreach:Weight $W_i$ do
    $z_{\text{tmp},1} \leftarrow z$, $z_{\text{tmp},2} \leftarrow z$        // Cache internal state of z
    $\epsilon \leftarrow \max\left(1, |W_i|\right)\sqrt{\epsilon_{\text{min}}}$        // Calculate $\epsilon$
    For $j = 1$ to $g$ do
        $\xi \leftarrow z_{\text{tmp},1}(x_j, W), \hat{\xi} \leftarrow z_{\text{tmp},2}(x_j, W - h_\epsilon)$
        For $o = 1$ to $q$ do
            $\hat{J}_{W(jm+o,i)} \leftarrow \frac{\xi_\nu - \hat{\xi}_\nu}{\epsilon}$        Backward difference
Return: $\hat{J}_W$
```

$z(x, W) \in \mathbb{R}^q$:	Dynamical system output
$x \in \mathbb{R}^{g \times n}$:	Inputs
$W \in \mathbb{R}^\nu$:	Weights

Table 3.1: Algorithm to numerically calculate the approximated Jacobian matrix.

[152], it is sufficient to calculate the partial derivatives of the systems output instead of the error function.

As the analytic calculation would result in extensive computations, it is numerical approximated using a difference equation, rather than a differential one as solution of the equation

$$\hat{J}_w\big(z(x, w)\big) = \begin{bmatrix} J_0 \\ J_1 \\ \vdots \\ J_k \end{bmatrix}^T, \quad J_i = \frac{z(x, w) - z(x, w - \zeta\varepsilon(w_i))}{\varepsilon(w_i)}, \quad zeta_p = \begin{cases} 1, & p = i \\ 0, & p \neq i \end{cases}, \tag{3.2}$$

where $\hat{J}_w$ is the approximated Jacobian matrix, ζ is the step size and ϵ is the machine precision. ϵ, has to be calculated for each pass as the solution of the equation

$$\epsilon(w_i) = \max\left(1, |w_i|\right)\sqrt{\epsilon_{\text{min}}}, \tag{3.3}$$

where w is a vector, containing the weights, and ϵ_{min} is the implementation data type, double precision in this implementation. The complete calculation of the Jacobian matrix can be found in Table 3.1.

As there exists no way of determining the ideal network architecture, for each system 10 different architectures were tested. All were defined by three layers, consisting of an input layer, one hidden layer and the output layer. The total number of neurons were within the neighbourhood of 10 neurons. The connections between those neurons and the delay for the connections were randomly varied. By choosing those parameters on a random basis, no knowledge about the process or its dynamics were intentionally included, making the experiments comparable. Furthermore, all the weights within the general dynamic neural networks were randomly initialized. For the evaluation, the neural network, yielding the best numerical results was chosen. An example neural network structure can be found in Figure 3.2. As the system to control has two outputs, the neural network has two input neurons and consists of 9 neurons in total. The neurons in the second layer feed information back to the first layer with a time delay of 1. The three output neurons correspond to the three parameters of the PID controller.

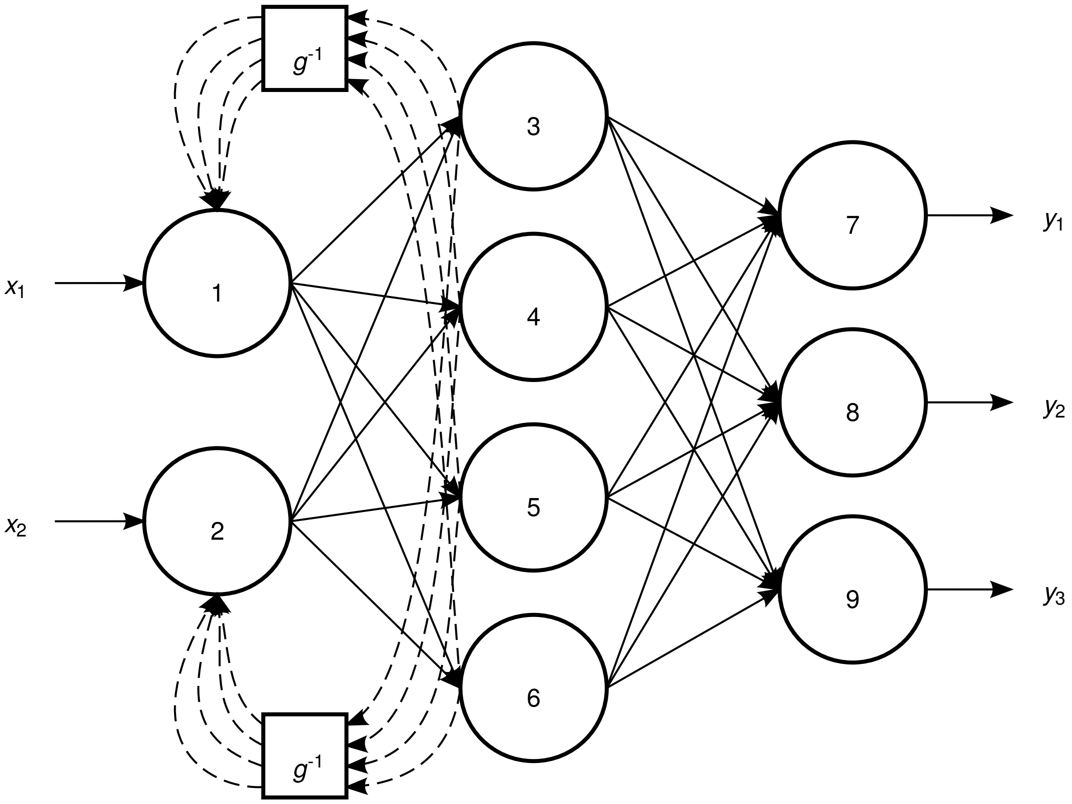


Figure 3.2: The General dynamic neural network, used to control the inverted pendulum.

3.2 Simulation and experiments

Training procedure: One of the most important decisions, when using machine learning algorithms, is related to the training data. Although the algorithms are supposed to be able to generalize from known data to unknown data, this capability is limited to some extent. For most algorithms, a specific phenomena has to be at least included twice in the training data to be learned and not be regarded as outlier. Geometrically speaking, the algorithm can not know about a specific subspace if it is not spanned by the training data. Therefore it will fail to identify it correctly in a test case. When confronted with a control system, the only way to interact with the system and collect information about it, is to apply an excitation signal. Such a signal will cause the system to assume different states, in order for the controller to learn about the system dynamics. The challenge is to find a signal that extracts as much information about the system as possible, but at the same time does not violate the physical constraints of the system.

In theory, a Dirac signal would immediately lead to the systems transfer function, allowing to design an appropriate controller. However, this proves difficult in real-world scenarios. Several excitation signals have been defined to excite control systems in order to collect information. Some of the most popular are: constant signals, impulse signals, step signals, rectangular signals or pseudo-random binary signals. All these signals have advantages and disadvantages. An overview can be found in [153]. In this thesis, four different test systems will be evaluated, that resemble the most common control problems. Each system has a different transfer functions and representing a different control task, including non-linearity and chaotic behaviour. Therefore the most general excitation signal should be used for all of them to ensure extracting the most information without the need to specifically adapt the excitation signal. As the pseudo-random binary signal (PRBS) excites all frequencies equally well, it is the most appropriate candidate. Other positive properties of PRBS signals are that they are deterministic and can easily be generated using linear feedback shift registers. These registers consist of several flip-flops that are arranged in a way such that the output of the former one serves as input for the next and an additional XOR block to the first flip-flop. All flip-flops share the same clock. Figure 3.3 shows such a linear feedback shift register with four states. Once the initial state is reached again, the shift register produces the same output periodically. In principle any state could be used to initialize the shift register, except for the all-zero state, because the shift register would then only be able to produce zeros. Table 3.2 shows an example output cycle, created by the four-state shift register from Figure 3.3 with an initial state of 1111. The final signal will by cyclic, as the internal state of the shift register is set back to its initial state of 1111 after the first 15 cycles. For the four shift register example, the signal would have a length—or maximum period— of 15, according to the equation

$$l_{max} = 2^{n_{register}} - 1, \tag{3.4}$$

where $n_{register}$ is the number of shift registers. Although the PRBS excites all frequencies equally well, this signal does not suffice. To collect a sufficient amount of information about the systems gain, the PRBS signal should have an ample holding time. This is the maximum number of 1's in a row and it naturally equal to the number of register. To prolong the holding time, the output of the register can be taken κ-times per clock cycle, with $\kappa \in \mathbb{N}$. To create a signal with a maximum holding time $t_{max} = 100$, the two parameters $n_{register}$ and κ have to be chosen accordingly. One could, for example, choose $\kappa = 1$ and $n_{register} = 100$,

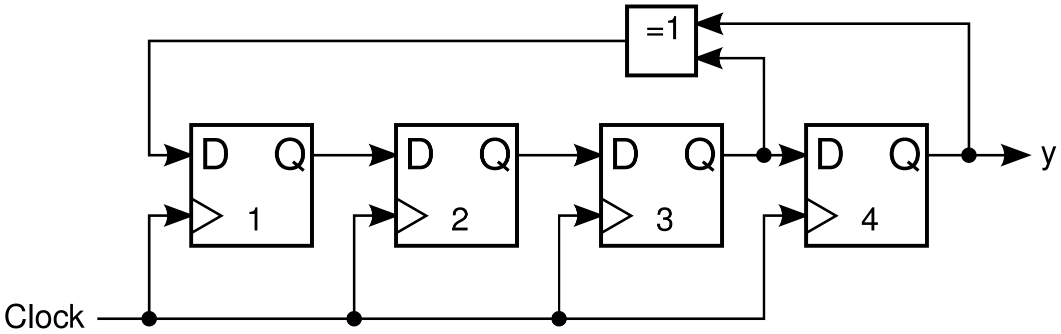


Figure 3.3: A linear feedback shift register with four states.

Table 3.2: One period in the linear feedback shift register from Figure 3.3.

Clock	State	Output y	Clock	State	Output y
0	1111	1	8	1100	0
1	0111	1	9	0110	0
2	0011	1	10	1011	1
3	0001	1	11	0101	1
4	1000	0	12	1010	0
5	0100	0	13	1101	1
6	0010	0	14	1110	0
7	1001	1	(15	1111	1)

but this would result in $2^{100}-1$ states, which is an unreasonable amount to be stored in the computers RAM. Another set of parameters would be to choose $n_{register} = 4$ and $\kappa = 25$, resulting in only 15 states. In addition, the first signal would have a length of $l_{max} * \kappa = 1023$, while the second would have a length of $l_{max} * \kappa = 375$. To reach an equivalent signal length of ≈ 1000, the second signal would have to be repeated nearly three times, resulting in redundant information. This can be interpreted as a simple optimization problem, which can easily be solved. For the given example of a signal with desired holding time $t_{max} = 100$ and a desired signal length of ≈ 1000 the ideal parameters are $n_{register} = 6$ and $\kappa = 17$, resulting in 63 states and a signal length of 1071, of which the first 1000 values can be used. The corresponding signal can be seen in Figure 3.4.

To find these parameters within reasonable time, the first 64 primitive polynomials, describing the relationship between the number of shift registers and the signal length can be stored to speed up the computation. As this results in a maximum signal length of $2^{64}-1$, which is the maximum array size in 64 bit computer architectures, these polynomials are sufficient.

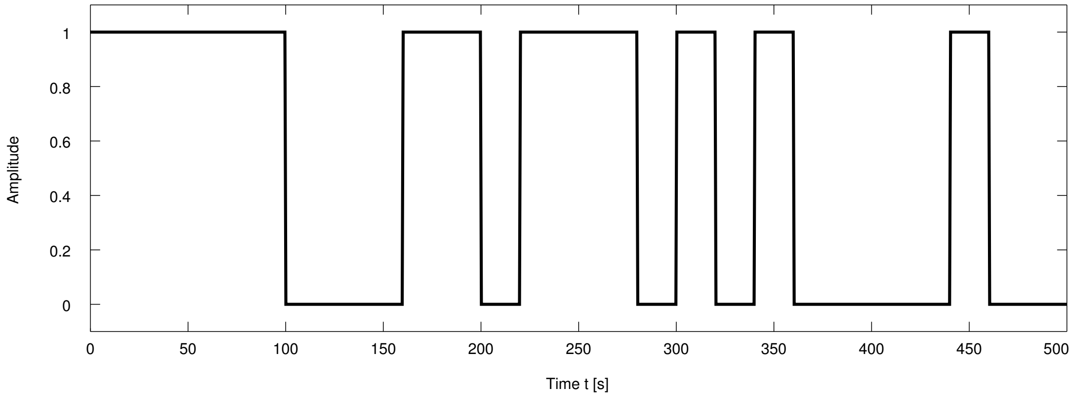


Figure 3.4: A PRBS with a maximum hold time of 100 seconds.

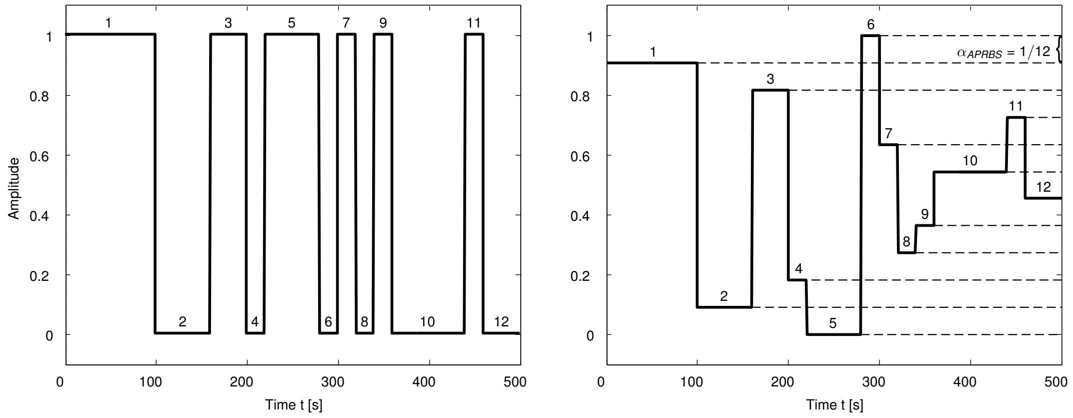


Figure 3.5: Transformation of a PRBS (left) to an APRBS (right) based on the amount of intervals.

As the PRBS signal only varies between the values of 0 and 1, it is not suitable for exciting nonlinear systems in a way that it is possible to collect all relevant information about the systems behaviour. This can be ensured by changing the PRBS in a way that it covers the whole range of possible input amplitudes. This is called an amplitude modulated PRBS (APRBS) and it has been shown to excite nonlinear systems appropriately [154]. To uniformly distribute the amplitudes between the desired maximum and minimum of the APRBS, the step size α between each amplitude step is introduced. It is determined by dividing the amplitude range by the amount of intervals of the PRBS. By transforming the PRBS into an APRBS, efficient training signals for the general dynamic neural networks can be generated. Figure 3.5 shows the transformation of a PRBS into a APRBS signal, dependent on the step size $\alpha_{APRBS} = \frac{1}{12}$ as a result of the number of intervals.

Experimental design: To find the appropriate parameters for the standard PID controller, several approaches were implemented. In a first iteration, a genetic algorithm with a pop-

ulation size and a number of generations of 1000 has been implemented to find the best parameters. Genetic algorithms have been shown to successfully find PID controller parameters for complex systems [83, 84] and can therefore be considered to be a competitive choice. The genetic algorithm was executed 100 times, resulting in 10^8 evaluated parameter sets. Furthermore, all systems were implemented and the MATLAB Control System Toolbox, as well as classical PID tuning approaches, e.g. Ziegler-Nichols were used. The resulting parameters were additionally verified by using a grid search approach to ensure they were the best parameters within their neighbourhood.
For evaluating the different PID tuning approaches in total four different control systems were implemented and considered. These systems represent the most common challenges in control theory and therefore provide a comprehensive overview. The systems are: a nonlinear two-tank system [155], an inverted pendulum on a cart [156], a time-invariant system with a non neglectable time delay [157] and a chaotic fluidsystem [158]. The systems will be introduced in the next section.

All four systems have been tested in 4 different configurations. First, the system was tested without any further modifications. Then noise was included to simulate real-world conditions, i.e. noisy sensor measurements.The noise was white Gaussian noise and calculated with a signal to noise ration $SNR = 20dB$ as solution of the equation

$$\mathrm{SNR} = 10\log_{10}\left(\frac{z^T z}{\beta^T \beta}\right)\mathrm{dB}, \tag{3.5}$$

where z is the systems output and β is the noise signal. The noise was then added to the systems output. To test all implemented control approaches for robustness, the systems were disturbed. Each disturbance was increased in strength, until all but one controller failed to stabilize the system. It is important to note that none of the controllers had experience with this sort of disturbance, as they were not included in the differential equations, the parameter tuning or training procedure. These disturbed runs were also run with and without noise, resulting in a total of 16 different test scenarios. To ensure the stability of the experiments and to minimize the possibility of numerical effects, all experiments were run for 30 independent runs with different seeds for the random number generator. The differential equations were solved using an ode45 solver. As this solver is a non-stiff solver, the control approaches would have constant access to the system, which is not feasible. To furthermore simulate real-world conditions, every controller can only adapt its control signal every 0.01s. This constraint is based on the sample time of a real sensor, which cannot provide a continuous flow of sensor data but only sample at a specific rate, based on its sample rate. These limitations are realized by repeatedly running the solver within this sample rate, while adapting the new starting conditions for each new run. Thereby a quasi-fixed step size of 0.01s is ensured. During this time, the controller output is kept constant.

3.2.1 Non-linear two-tank system

One of the most considered benchmark system in control theory is the nonlinear two-tank system [159]. The controller has access to the input voltage $u(t)$, which runs a pump with a pump constant k_P. The goal for the controller is to keep the water level in the second tank $z(t) = x_2(t)$ at a defined level. Both tanks are cylindrical shaped with the cross sectional areas of A_1 and A_2, respectively. Furthermore the outlets are defined by their cross sectional areas of A_{o1} and A_{o2}. The system is characterized by the differential equations

$$\begin{aligned} \dot{x}_1(t) &= -\frac{A_{o1}}{A_1}\sqrt{2k_{gravitation}x_1(t)} + \frac{k_{pump}}{A_1}u(t)\,, \\ \dot{x}_2(t) &= \frac{A_{o1}}{A_2}\sqrt{2k_{gravitation}x_1(t)} - \frac{A_{o2}}{A_2}\sqrt{2k_{gravitation}x_2(t)} = z(t), \end{aligned} \tag{3.6}$$

and the specific values for the constants can be found in Table 3.3. It was chosen to be the first system for comparison as it is a standard benchmark system that is widely used to evaluate controllers for their capability of controlling nonlinear systems. real-world examples for two-tank systems include, but are not limited to, bio-reactors, filtration system or nuclear power plants [155]. For such systems it is of enormous importance to keep a steady water level, as a failing cooling water level might, for example, lead to non sufficient cooling which could lead to explosions. This kind of system can be seen as widely used and is therefore an important candidate for control. Already existing approaches for control include neural networks [160], adaptive output feedback [161] and backstepping [155]. For this work, a backstepping algorithm, which is an adaptive controller [162], was implemented as a comparative control approach for the PID tuner.

To further evaluate the controllers robustness, the system was disturbed between $t = 20$s and $t = 40$s. In this timespan the input to the system was set to 0, regardless of the controllers actions. This problem would correspond either to a leakage in the inflow or to a stuck ventilate, given the input was already at 0. Such a problem can widely occur due to the susceptibility of mechanical components and can be considered a typical incidence.

Table 3.3: Physical parameters of the two-tank system.

Parameter	Symbol	Value	Unit
Tank 1, 2 cross-sectional areas	A_1, A_2	15.38	cm^2
Tank 1, 2 orifice cross-sectional areas	A_{o1}, A_{o2}	0.1781	cm^2
Pump constant	k_{pump}	4.6	cm^3/Vs
Gravitational constant	$k_{gravitation}$	980	cm/s^2

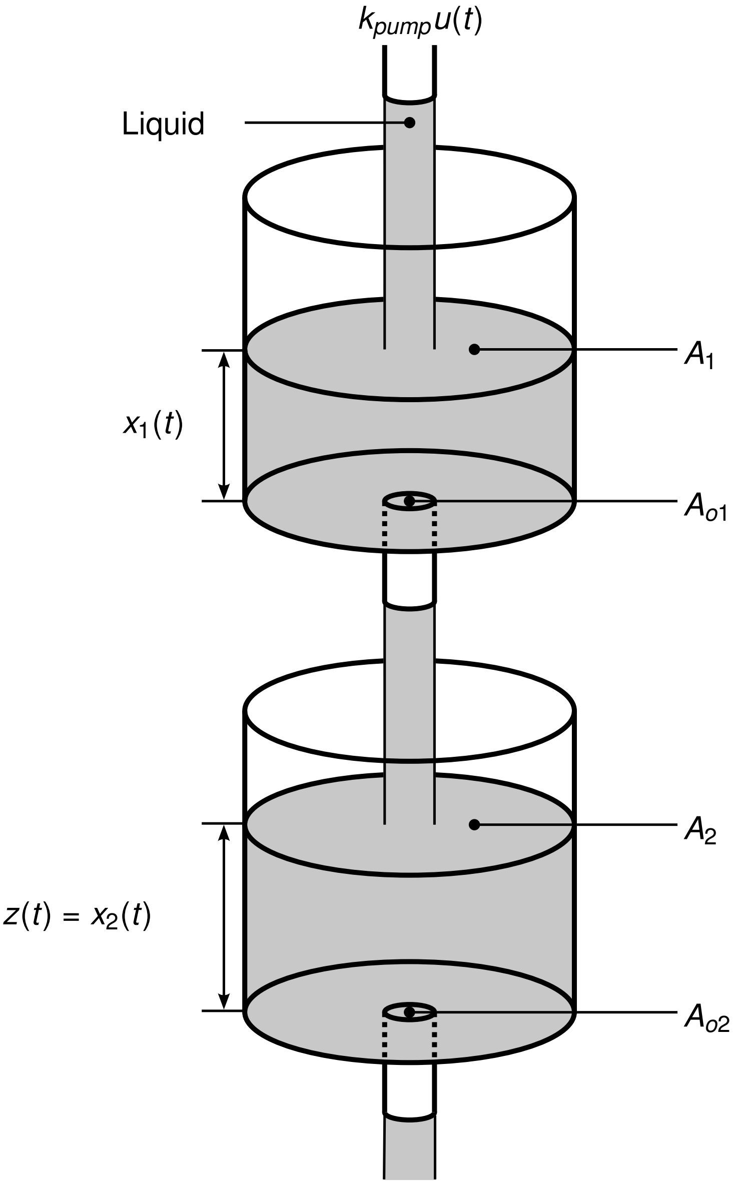


Figure 3.6: The nonlinear two tank system.

3.2.2 Inverted pendulum on a cart

The second test system is an inverted pendulum on a cart. This system is characterized by its nonlinearity and unstable behaviour. The specific control task is to stabilize the inverted pendulum at its unstable equilibrium, while the cart is not allowed to exceed a range of ±0.5m. The dynamics of the system can be expressed as a system of differential

Table 3.4: Physical parameters of the inverted pendulum on a cart.

Parameter	Symbol	Value	Unit
Cart mass	ρ_c	1	kg
Pendulum mass	ρ_p	0.1	kg
Pendulum length to mass center	l_p	0.3	m
Gravitational constant	$k_{gravitation}$	9.8	m/s^2

equations which can be seen in equation

$$\begin{aligned}
\dot{x}_1(t) &= x_2\,, \\
\dot{x}_2(t) &= \frac{-\rho_p k_{gravitation}\cos(x_3)\sin(x_3) + \rho_p l_p \sin(x_3) x_4^2 + F_x}{\rho_c + \rho_p \sin^2(x_3)}, \\
\dot{x}_3(t) &= x_4\,, \\
\dot{x}_4(t) &= \frac{-\rho_p l_p \cos(x_3)\sin(x_3) x_4^2 - \cos(x_3) F_x}{\rho_c l_p + \rho_p l_p \sin^2(x_3)} + \frac{(\rho_c + \rho_p) k_{gravitation}\sin(x_3)}{\rho_c l_p + \rho_p l_p \sin^2(x_3)},
\end{aligned} \tag{3.7}$$

where x_1 and x_2 denote the position x and the speed $\dot{x}$ of the cart, while x_3 and x_4 denote ϕ and its angular velocity $\dot{\phi}$. In the differential equations, the dependency of the states x_i of t it omitted, such that $x_i = x_i(t)$. The exact physical parameters for the system can be found in Table 3.4. It is furthermore a well known and extensively researched problem in control theory, making it a valuable test system for this comprehensive overview. A practical application for this type of control problem can be the initial stage of flight for a missile launch. Very common approaches to control such unstable and nonlinear problem are LQ-regulators as a represent of adaptive control [163, 164, 165] or a double PID controller approach [163]. In this work, both approaches were implemented as a comparison.

In order to test all control approaches to robustness, once the system has settled, a force of 8.5N has been applied to the tip of the pendulum. This is comparable to a strong and unexpected wind condition during the launch of a missile.

3.2.3 System with non-neglectable time delay

The next control system is a first order linear time invariant (LTI) system with a non-neglectable time delay. While time delay is a serious and challenging problem in certain applications, e.g. heating systems or production engineering systems like milling or cutting, it is often not considered while designing controllers [166]. If not appropriately taken care of, it results in decreased performance or even instability. Most common controllers for these types of systems are fractional order PID controller [167] or a smith-predictor [168], which is a member of the predictive control approach [169]. A linear plant with

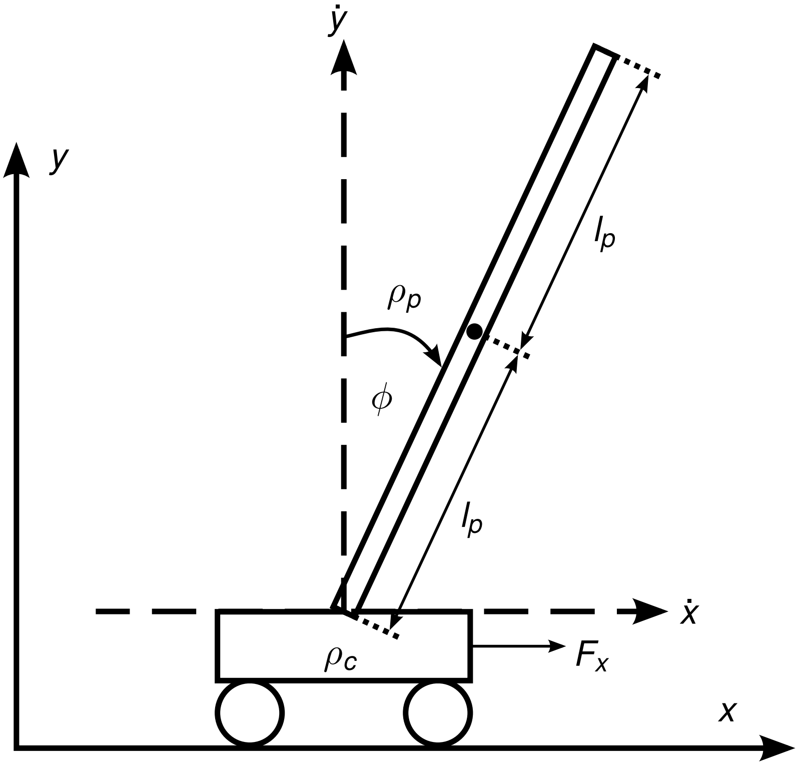


Figure 3.7: The inverted pendulum on a cart.

non-neglectable time delay is characterized by a gain K and its input delay T_D. The time constant T is the time the system needs to reach the amplitude of $K_{63} = K(1 - \frac{1}{e}) \approx 0.63K$ as response to a unit step after the input delay of T_D. The system at hand is described by equation

$$F(s) = \frac{K}{Ts + 1} e^{-T_D s} \bullet\!\!-\!\!\circ\, \dot{x}_1 = \frac{1}{T}\Big(Ku(t - T_D) - x_1\Big), \tag{3.8}$$

and its parameters are: $K = 0.4$, $T = 0.9$, $T_D = 1.8$ [157].

This system is disturbed by a (dimensionless) disturbance of -5 between $t = 50$s and $t = 75$s. Such a disturbance could correspond to a temporary blockage in a heating system, the feed rate time in a milling system [170] or a lag in a communications network.

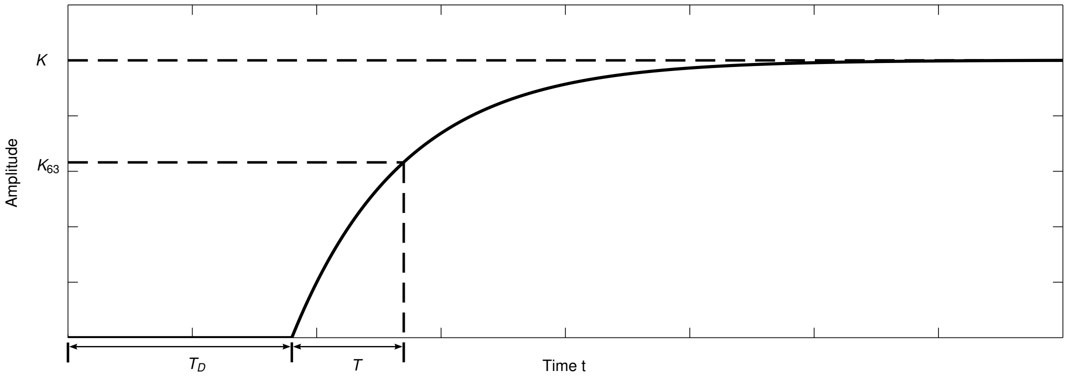


Figure 3.8: Unit step response of a first order linear time-invariant system with input delay.

3.2.4 Chaotic fluid system

For the fourth system a chaotic thermal convection loop, which is described by the equation

$$\begin{aligned}\dot{x}_1(t) &= k_{prandtl}(x_2 - x_1)\,,\\ \dot{x}_2(t) &= x_1 - x_2 - x_3(x_1 + k_{rayleigh}),\\ \dot{x}_3(t) &= x_1x_2 + k_{rayleigh}(x_1 + x_2) - x_3 - u,\end{aligned} \tag{3.9}$$

was chosen. This system is characterizing a fluid within a torus. While the lower half of the torus is enclosed by an electric heating system, the upper half is cooled down using a water cooling. The goal is to achieve a constant flow through the inner torus. Chaotic behaviour is common in real systems. It occurs where a system is globally bounded but locally unstable [171], for example in most fluids or due to gravitation. As chaotic behaviour can lead to vibrations, oscillations and failure in systems it is a very important problem in control theory and therefore a good test scenario. Due to the lack of sufficient mathematical models for real chaotic systems a closed loop control is a desirable approach [172]. For the example of a chaotic thermal convection loop, nonlinear feedback controllers [158] and backstepping [173, 174, 175] are two established adaptive control approaches. In this work, a backstepping control approach was implemented for comparison.

During the experiments, a disturbance of -100W between t = 5s and t = 5.5s has been applied to test the control approaches for robustness. For the system at hand, this can be interpreted as a change in the cooling water temperature.

3.3 Results

To make all the different approaches comparable, a common error measure has to be found. The most common error measure in machine learning is the root mean squared

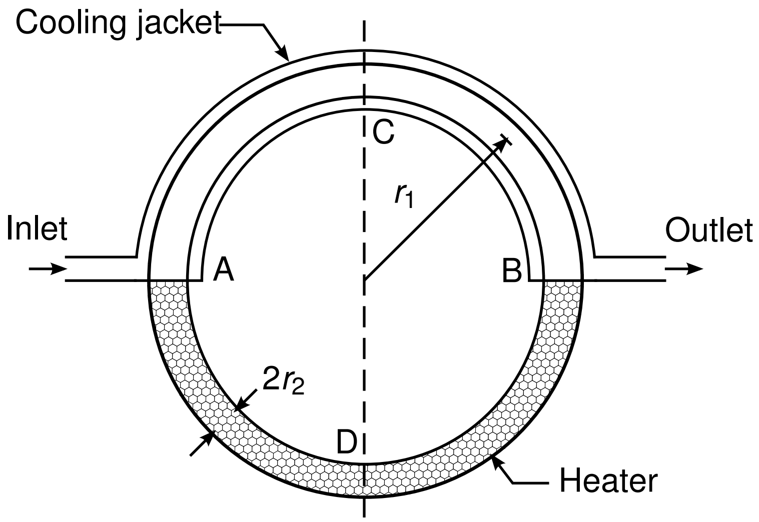


Figure 3.9: Chaotic thermal convection loop.

Table 3.5: Physical parameters of the thermal convection loop.

Parameter	Symbol	Value	Unit
Prandtl constant	$k_{prandtl}$	10	-
Rayleigh number (substituted)	$k_{rayleigh}$	6	-
Convection loop radius	r_1	38	cm
Tube radius	r_2	1.5	cm

error (RMSE), which is defined by the equation

$$\text{RMSD} = \sqrt{\frac{(z - z^*)^T(z - z^*)}{n}}, \tag{3.10}$$

where z is the systems output, z^* is the ideal output and n is the length of the error vector, or number of samples in the experiment. This measure has the advantage of considering the performance over the whole experiment and in addition it can be seen as an average over the classical error measures that are used for PID control problems, e.g. rise time, overshoot, settling time. It therefore provides an ideal error measure for an approach, combining classical PID-control technology and modern machine learning algorithms. For all four introduced systems, the standard PID controller will be compared to one model-based state of the art control approach and the newly suggested adaptive PID controller. The systems will be controlled with and without noise as well as with and without disturbance, resulting in 16 test scenarios in total.

The results for all experiments can be found in Table 3.6. For each experiment, the mean and the variance of the root mean square error are provided. The controller with the best performance is marked bold. In the experiments, where the variance is zero, the experimental results for all 30 independent runs were identical, due to the absence of noise or randomness.

3.3.1 Non-linear two-tank system

The results for the first system can be seen in Figure 3.10. For this system, a standard PID controller with the values $K_P = 3.65$, $K_D = -2$ and $K_I = 0.4$, a backstepping algorithm [155] and the adaptive PID controller were implemented. Depicted is the control performance of all three controllers for the disturbed system without noise. The water levels in both tanks were initialized to 5 and the set point during the test sequence was created, using a ARPBS signal with a maximum holding time of 30. For the sake of visibility, the example without noise was chosen.

It can be seen that all three algorithms are able to stabilize the system after initialization. While the standard PID controller approaches the target liquid level a bit faster than the two other approaches, it also overshoots, once the desired level is achieved. The adaptive PID controller is almost as fast as the standard PID controller but has almost no overshoot, quickly stabilizing the system at the desired level. The backstepping algorithm also does not overshoot, but takes considerably more time to reach the correct liquid level.

At the time $t = 20$s, the disturbance is introduced. For the next 20s all three algorithms fail to follow the set point, as they lose all influence on the system. After the disturbance, at $t = 40$s, the PID-controller shows a significant overshoot, due to the integrator of the controller acquiring a big error during the disturbance. For the rest of the experiment, the PID controller is able to follow the set point, but it consistently overshoots, resulting in a poor overall performance. The backstepping is more steady than the standard PID controller, but lacks a fast response, resulting in a poor rising time and therefore a big control error. The new suggested adaptive PID controller combines the fast response of the standard PID controller with the smoothing effect of the backstepping, resulting in a very accurate and fast performance. As the set point in the simulation changes, it can be observed how the neural network adapts the parameters, mainly the K_P. This consistent tuning and linearization around the actual systems state turns out to result in a good control result. Due to the convergence and the reduced overshoot, the overall controller output is significantly smaller for the adaptive PID controller, resulting in less consumed pump power. In the overall experiment, the adaptive PID controller has a 14.9% smaller RMSD than the standard PID controller and a 23.6% smaller error than the backstepping approach. It demonstrates this superior behaviour for all four tests, conducted on the two-tank system.

Table 3.6: Control results for the four benchmark systems over 30 independent runs.

Control Benchmark		**RMSE on test data over 50 independent runs**			
Disturbance		-	-	✓	✓
SNR		-	20 dB	-	20 dB
Two-tank system					
Mean	Standard PID	$9.5 \cdot 10^{-1}$	$9.9 \cdot 10^{-1}$	$1.0 \cdot 10^{0}$	$1.1 \cdot 10^{0}$
	Backstepping	$1.1 \cdot 10^{0}$	$1.1 \cdot 10^{0}$	$1.2 \cdot 10^{0}$	$1.1 \cdot 10^{0}$
	Adaptive PID	$\mathbf{7.4 \cdot 10^{-1}}$	$\mathbf{8.6 \cdot 10^{-1}}$	$\mathbf{8.4 \cdot 10^{-1}}$	$\mathbf{1.0 \cdot 10^{0}}$
Variance	Standard PID	$2.5 \cdot 10^{-3}$	$2.3 \cdot 10^{-3}$	$5.2 \cdot 10^{-3}$	$6.3 \cdot 10^{-3}$
	Backstepping	$1.6 \cdot 10^{-3}$	$1.8 \cdot 10^{-3}$	$3.6 \cdot 10^{-3}$	$2.4 \cdot 10^{-3}$
	Adaptive PID	$3.6 \cdot 10^{-3}$	$4.0 \cdot 10^{-3}$	$5.6 \cdot 10^{-3}$	$4.6 \cdot 10^{-2}$
LTI system with input delay					
Mean	Standard PID	$2.2 \cdot 10^{-1}$	$2.3 \cdot 10^{-1}$	$3.6 \cdot 10^{-1}$	$3.8 \cdot 10^{-1}$
	Smith predictor	$1.8 \cdot 10^{-1}$	$1.9 \cdot 10^{-1}$	$\mathbf{1.9 \cdot 10^{-1}}$	$\mathbf{2.0 \cdot 10^{-1}}$
	Adaptive PID	$\mathbf{1.3 \cdot 10^{-1}}$	$\mathbf{1.5 \cdot 10^{-1}}$	$2.6 \cdot 10^{-1}$	$2.8 \cdot 10^{-1}$
Variance	Standard PID	$6.7 \cdot 10^{-4}$	$6.1 \cdot 10^{-4}$	$4.0 \cdot 10^{-4}$	$3.2 \cdot 10^{-4}$
	Smith predictor	$5.6 \cdot 10^{-4}$	$4.7 \cdot 10^{-4}$	$4.9 \cdot 10^{-4}$	$3.2 \cdot 10^{-4}$
	Adaptive PID	$6.0 \cdot 10^{-4}$	$5.3 \cdot 10^{-4}$	$6.8 \cdot 10^{-4}$	$3.4 \cdot 10^{-4}$
Inverted pendulum					
Mean	Standard PID	$3.5 \cdot 10^{-2}$	$3.6 \cdot 10^{-2}$	$1.4 \cdot 10^{2}$	$1.4 \cdot 10^{2}$
	LQ regulator	$5.2 \cdot 10^{-2}$	$5.3 \cdot 10^{-2}$	$1.4 \cdot 10^{2}$	$1.4 \cdot 10^{2}$
	Adaptive PID	$\mathbf{3.4 \cdot 10^{-2}}$	$\mathbf{1.8 \cdot 10^{-2}}$	$\mathbf{9.0 \cdot 10^{-2}}$	$\mathbf{2.7 \cdot 10^{-2}}$
Variance	Standard PID	0	$1.1 \cdot 10^{-7}$	0	$2.3 \cdot 10^{-3}$
	LQ regulator	0	$3.3 \cdot 10^{-9}$	0	$2.2 \cdot 10^{-3}$
	Adaptive PID	$4.0 \cdot 10^{-4}$	$2.6 \cdot 10^{-4}$	$4.9 \cdot 10^{-2}$	$7.3 \cdot 10^{-4}$
Thermal convection loop					
Mean	Standard PID	$2.4 \cdot 10^{-1}$	$1.6 \cdot 10^{0}$	$1.3 \cdot 10^{1}$	$4.2 \cdot 10^{0}$
	Backstepping	$2.6 \cdot 10^{-1}$	$\mathbf{8.9 \cdot 10^{-1}}$	$9.8 \cdot 10^{0}$	$9.8 \cdot 10^{0}$
	Adaptive PID	$\mathbf{2.3 \cdot 10^{-1}}$	$9.0 \cdot 10^{-1}$	$\mathbf{1.9 \cdot 10^{0}}$	$\mathbf{1.7 \cdot 10^{0}}$
Variance	Standard PD	0	$2.0 \cdot 10^{0}$	0	$1.4 \cdot 10^{0}$
	Backstepping	0	$1.3 \cdot 10^{-4}$	0	$1.4 \cdot 10^{-5}$
	Adaptive PID	$1.2 \cdot 10^{-5}$	$3.1 \cdot 10^{-4}$	$1.6 \cdot 10^{-1}$	$2.2 \cdot 10^{-1}$

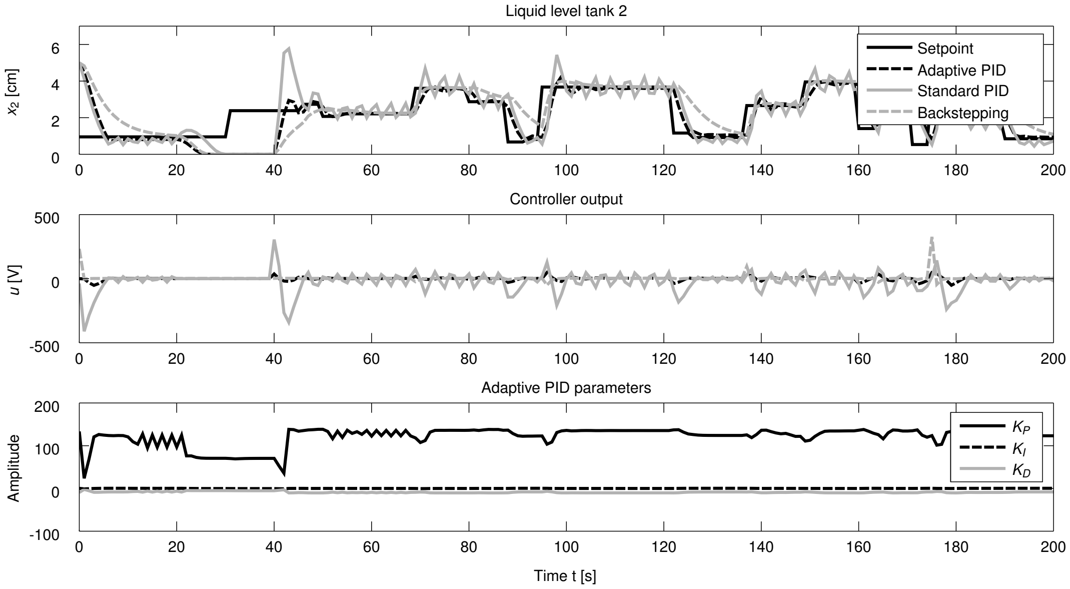


Figure 3.10: Control results for the two-tank system without noise and with disturbance between t = 20s and t = 40s.

3.3.2 Inverted pendulum on a cart

For the inverted pendulum on a cart, a standard PID controller stack [156], a LQ regulator [163] and the adaptive PID controller were evaluated. As this system is a single-input, multiple output system, two adaptive PID controllers are implemented, similar to the stacked PID approach [156]. The values for the two standard PID controllers are taken from [156]. For the PID controller, responsible for the position the values are $K_P = -2.4, K_D = -0.75$ and $K_I = -1$ and for the angle $K_P = 25, K_D = 3$ and $K_I = 15$. It has to be noted that the PID controller for the angle has a positive feedback loop. The system was initialized with the cart at position $x = 0$ and the pendulum at an angle of $\phi = 1 rad$. As this system is unstable, it was the controllers goal to stabilize it at its equilibrium. The results for the system without noise and with a disturbance are shown in Figure 3.11.

All three approaches are able to stabilize the system in the beginning. While the LQ regulator has almost no overshoot, it takes the longest time to control the system in a way that the x position and the angle are correct. The standard PID controller is faster than the LQ regulator but has a bigger overshoot, due to its aggressive behaviour. The adaptive PID controller is the fastest of the three and has a smaller overshoot than the standard PID controller, resulting in the best initial performance.

At the time t = 10s, the system is disturbed with a force of 8.5N at the tip of the inverted pendulum for 0.5s. The value of 8.5N was determined by continuously increasing the force from 0N, until all but one controller fail. The disturbance was not included in the training data, so none of the controllers was familiar with it. It can be seen that both traditional control approaches, the standard PID controller and the LQ regulator, fail to stabilize the system after the disturbance is applied. The cart moves out of the limited range and the inverted pendulum falls over. Only the adaptive PID controller manages to regain balance within the restrictions of the system by adapting the PID parameters. It is interesting to notice that there is almost no change in the PID parameters for the angle but only in the ones for the PID controller, responsible for the position. This could indicate that the adaptive PID controller aims for a fast movement of the cart to first stabilize the inverted pendulum and then slowly moves it back to its set point. The graphs, showing the position and angle support this. During the whole experiment there is only a small controller output necessary, except for the stabilizing in the beginning and after the disturbance. As the traditional controllers fail to stabilize the system, the adaptive PID controller has a 99.9% smaller RMSD. It also demonstrates to be superior in all four test cases. This is especially remarkable, as the adaptive PID controller has only access to the set point and the current control error, while the LQ regulator has significantly more information about the system.

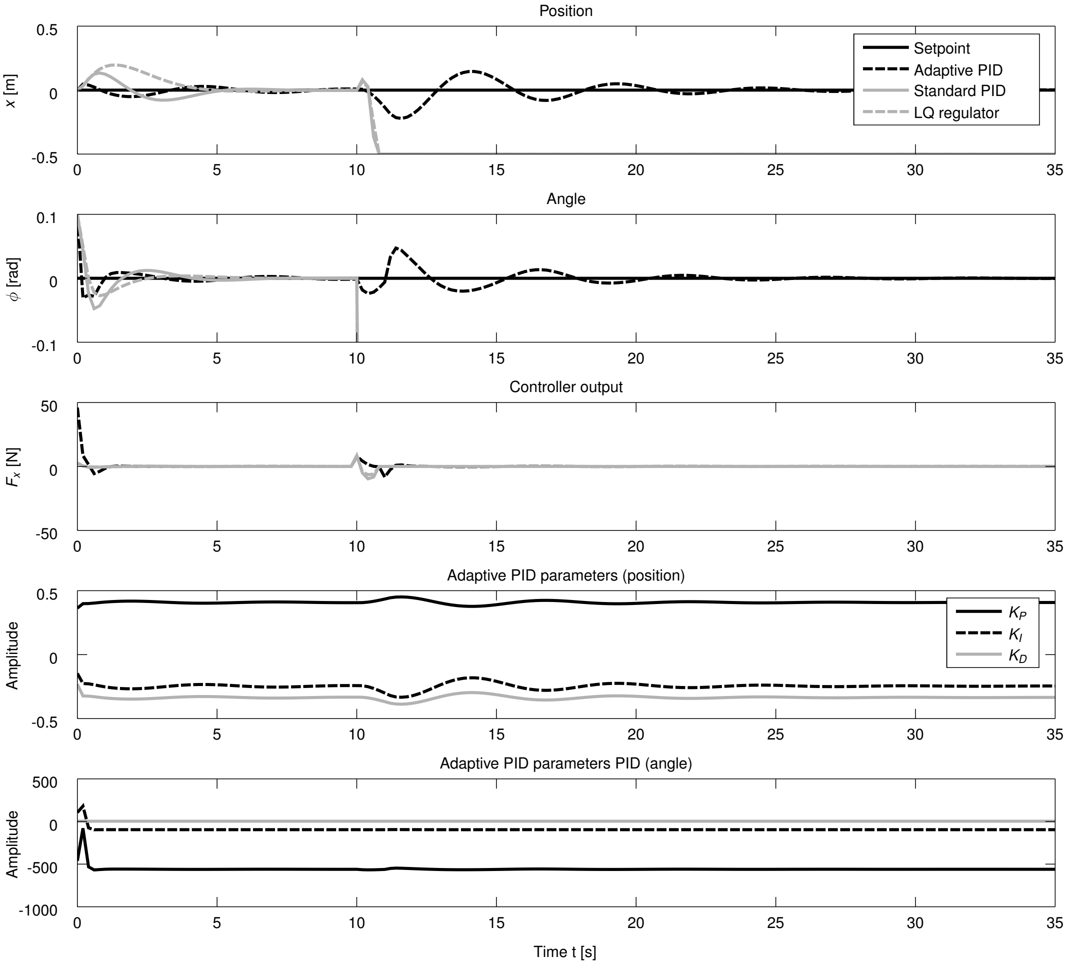


Figure 3.11: Control results for the inverted pendulum on a cart without noise but with disturbance at t = 10s.

3.3.3 System with non-neglectable time delay

The system with a non-neglectable time delay was controlled, using the standard PID controller with the parameters $K_P = 1.5, K_D = -0.1$ and $K_I = 0.7$, a smith predictor [176] and the adaptive PID controller. The test signal, i.e. the set point, was created, using an ARPBS signal, with a range of $[-1, 1]$ and a maximum holding time of 50. The control results with disturbance but without noise can be seen in Figure 3.12. Between the time t = 50s and t = 75s, a dimensionless disturbance of -5 was applied to the system to test all three controllers for robustness. Such a disturbance might correspond to a blockage in a fluid transport system.

Especially between the time $t = 65$ and $t = 70$, the influence of the time delay on the control performance can be inspected. While all controllers are eventually able to reach the set point, the smith predictor performs best. This result was to be expected, as the smith controller does have additional knowledge about the exact time delay, while the standard PID controller and the adaptive PID controller rely on the control error. However, the adaptive PID controller still performs better than the smith predictor for the two test scenarios without disturbance. It furthermore outperforms the standard PID controller by 31.1%. This is achieved by only a minimal parameter adaption. The parameters varied from 6.21918 to 6.2255 for K_P, from 1.1285 to 1.13003 for K_I and from -0.44297 to -0.441411 for K_D.

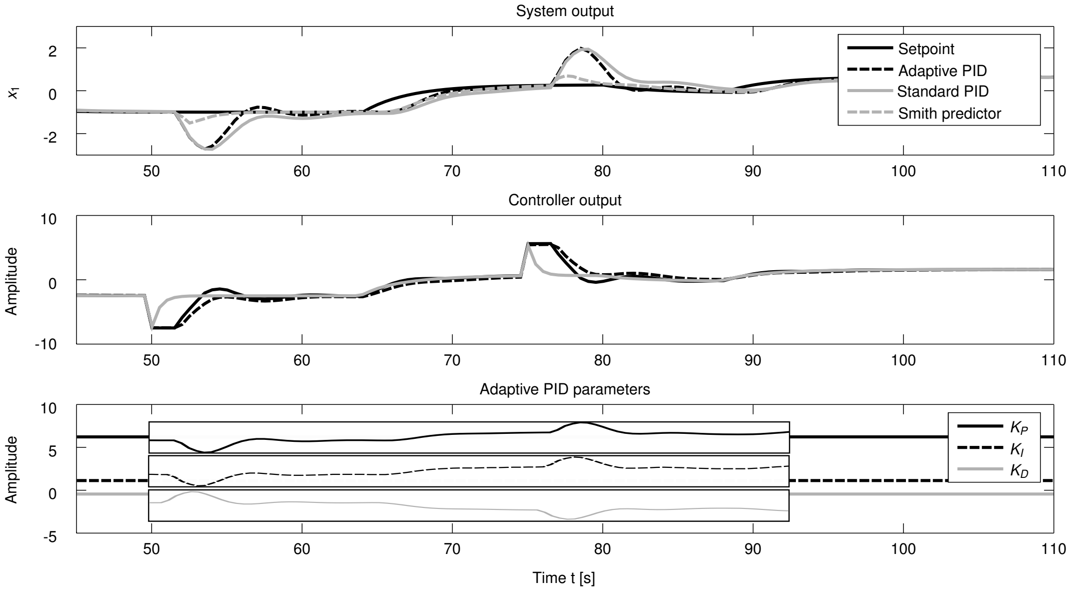


Figure 3.12: Control results for the system with a non-neglectable time delay without noise and with disturbance from $t = 50$s till $t = 75$s.

3.3.4 Chaotic fluid system

The last system to be evaluated is the chaotic fluid system. To control the system, three controllers—a standard PID controller, a backstepping algorithm and the adaptive PID controller—were implemented. The solution for the PID parameters, found by the genetic algorithm, got stuck at a value of $x_1 = -6$. This behaviour can be explained by investigating the differential equations. If the value of x_1 becomes equal to the negative Rayleigh number of 6, the term $x_3(x_1 - \beta)$ in the second equation becomes 0, canceling the influence of x_3 on the other two differential equations. However, the controller has only access to the x_3 and therefore gives up control, letting the system slowly but continuously drift away from the set point. This unusual behaviour is a result of the optimization target for the genetic algorithm. It evaluates the RMSE within the experimental time and tries to minimize it. As the experiments only run for a limited time, this result was the best within the neighbourhood of the parameters. The algorithm got stuck on a local, but not the global minimum, as it yielded the best numerically result, regardless of the fact that the controller did not control at all. After evaluation, this result was dropped and more classical PID tuning approaches,e.g. the Ziegler-Nichols approach and also the MATLAB Control System Toolbox, were evaluated. To further ensure the best solution, a brute force random search, using a fractal experimental design was tested. The final PID parameters were $K_P = 25.3$, $K_D = 8.9$ and $K_I = 0$.

The initial conditions for the system were $x_1 = x_2 = x_3 = 5$. As the chaotic fluid system is an unstable system, the goal for all three controllers was to stabilize the system at $x_1 = 0$. The control results for the disturbed system without noise can be seen in Figure 3.13. All three algorithms are capable of stabilizing the system within a very short time horizon. The backstepping has a very small overshoot and quickly approaches the set point, nevertheless having the highest control error until the steady state is reached. The standard PID controller is more aggressive, resulting in a higher overshoot but eventually a smaller time until convergence. The adaptive PID controller performs best, as it finds a good balance between a short rising time and overshooting. During the first seconds, the standard PID controller iterates the output between the maximum and the minimum with a continuously falling amplitude. The backstepping and the adaptive PID controller need considerable less power to stabilize the system.

Between the time $t = 5$s and $t = 5.5$s, the disturbance of -100W is applied, regardless of the controllers control actions. Again, the value of -100W was chosen, as all but one controller fail. The standard PID and the backstepping algorithm are not able to stabilize the system. However, the system does not become unstable, either. Both algorithms iterate between the maximum and the minimum controller output, resulting in a metastable process. Only the adaptive PID controller is capable of controlling the process and transfer the flow velocity back into its equilibrium. As the process is stabilized again, the controller output for the adaptive PID controller approaches 0, again. Similar to the final parameters for the standard PID controller, the K_I parameter for the adaptive PID controller stays almost 0 for the whole experiment, while K_P and K_D are adapted. It is interesting to note that the backstepping algorithm outperforms the adaptive PID controller for the experiment without disturbance but with noise as can be seen in Table 3.6. This can be explained by the fact that the backstepping algorithm was implemented, using the differential equations, therefore knowing the underlying, true structure of the problem. The adaptive PID controller on the other side can only act in a data-driven way and is therefore more prone to noisy measurements.

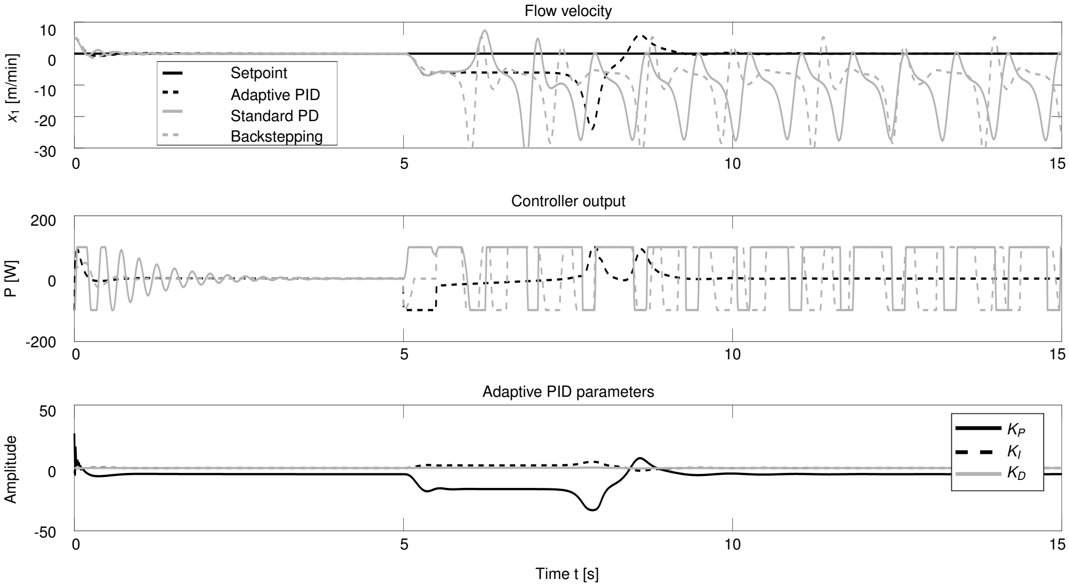


Figure 3.13: Control results for the chaotic fluid System without noise and with disturbance between t = 5s and t = 5.5s.

3.4 Internal dynamics vs. external dynamics

As this chapter suggests the usage of general dynamic neural networks for PID parameter tuning, the questions arises whether the general dynamic property actually yields a superior performance over more simple feed-forward neural networks. To investigate this question, a general dynamic neural network was compared with a static feed-forward neural network and with a static feed-forward neural network that was extended by an external dynamic. The general dynamic neural network was the eventually found to control the system and is depicted in Figure 3.2. For the static feed-forward neural network, the same network without the recurrent connections in the first layer was used. Furthermore, a feed-forward neural network with an external dynamic was implemented as an intermediate step between the general dynamic one and the static one [47]. As the neural network with an external dynamic achieves makes use of additional inputs to include the temporal structure of the data, this type of approach will grow in dimension for each additional system output. To address and evaluate this quality, the inverted pendulum on a cart was chosen as the benchmark system. This system is nonlinear and it has an unstable equilibrium at ϕ = 0 rad, making it hard to control. All three neural networks were repeatedly randomly initialized and tested for convergence and performance.

From Figure 3.14 it can be seen that the recurrent neural network that was suggested in this thesis has a significantly higher success rate than the other two approaches. While it stabilized the system with a 72.2% rate, the neural network with an external dynamic only

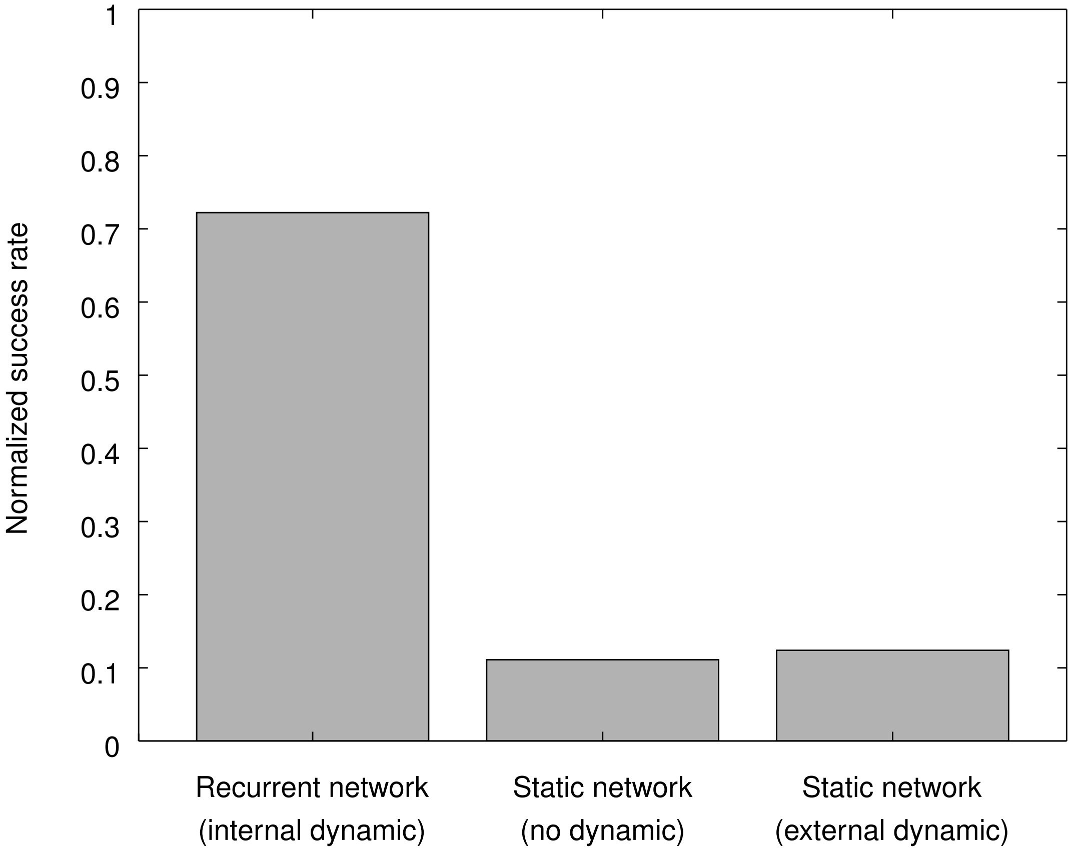


Figure 3.14: Comparison of the recurrent neural network, used for the inverted pendulum on a cart, with a feed-forward network and a feed-forward network with an external dynamic.

achieved a rate of 12.4% and the feed-forward network without external dynamic 11.1%. In addition, the control performance of the successful recurrent neural networks were significantly better than for the two approaches without an internal memory. The numeric values were $RMSD = 2.2 * 10^{-1}$ for the recurrent neural network, $RMSD = 4.0 * 10^{-1}$ for the neural network with an external dynamic and $RMSD = 6.8 * 10^{-1}$ for the simple feed-forward neural network. These results suggest that the internal memory, which is introduced into the neural network by recurrent and arbitrary connections do significantly improve the usability for control problems. The higher success rate indicates that the training process using such a neural network, will be considerably shorter as fewer random initializations have to be performed to find a stable solution.

3.5 Conclusions and contributions of this chapter

In this chapter we introduced the idea of using general dynamic neural networks for an adaptive PID controller and compared those machine learning based controllers with standard PID controllers and adaptive control approaches [177]. The adaptive PID controller proved to be superior in 13 out of 16 cases.

Via the extension by neural networks, the simple PID controller became a superior controller, even when compared to state of the art model-based controllers. This type of closed loop control approach does only require the same information that are provided to the standard PID controller. Such a data driven control approach promises to address key requirements in modern industry, such as disturbance rejection and adaption to changing system dynamics. It can be seen as a first step towards a machine learning based approach to adaptive stochastic control.

4 Closing the loop for industrial laser welding

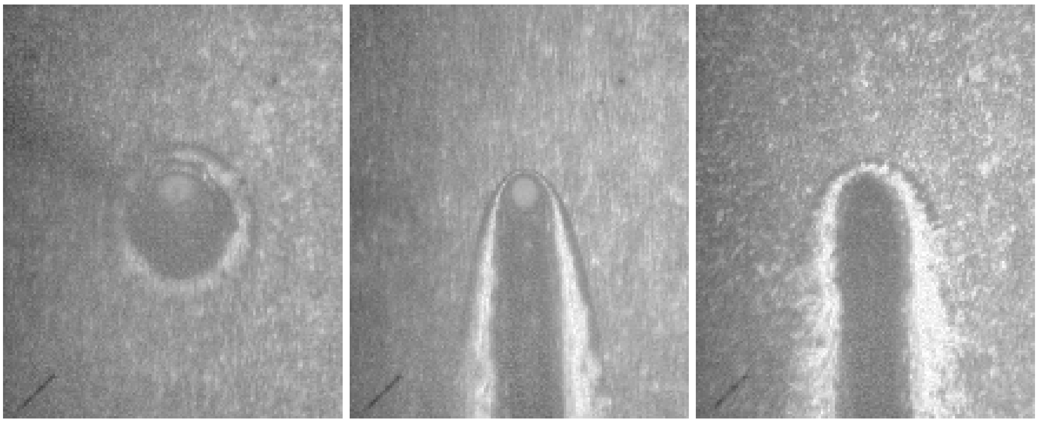
As shown in the last chapter, extending existing control approaches can lead to superior performance for the particular tasks. However, in industry there are several applications where traditional control approaches cannot be applied in the first place. This might either be due to a lack of sufficient mathematical models in traditional, model-based control theory or due to a lack of a sufficient feedback signal for data-driven closed loop control.

For this kind of process, a completely new control approach is required. One prominent example of such a process is laser welding and it will be used as an example of how to conceptualize such a new control architecture.

4.1 The laser welding data set

Machine learning algorithms are mainly dependent on the provided data and its quality. As the wrong choice of data might result in a biased learning result, it is essential to closely inspect and evaluate the data set before any learning occurs. The laser welding data set was created in an industrial laser welding environment to ensure real-world conditions. The data is structured into different processes (P), each one containing numerous welds (W). The processes are generated, using different experimental setup. For different processes, the focal point or the setup of the camera might differ. Each individual welding within a process contains between $1,500$ and $2,500$ images. For every weld there is a stream of image data, recorded by a high-speed camera in QCIF resolution (176×144) and the corresponding photo diode data available. Examples for the raw welding images can be inspected in Figure 4.1.

For the experiments, the metal sheets were arranged in an overlap position, similar to real car manufacturing processes. Similar to a car manufacturing, the welding pieces consisted of zinc-coated steel. During the recording of the data, several typical problems were simulated, e.g. the laser power was changed during the welding, the speed of the laser was altered, the metal sheets were polluted with grease or adhesive tape, etc. These are all typical problems that might occur in industry due to inattentiveness or errors in the programming of the laser. Every conducted weld was afterwards cut, treated with chemicals and labeled in accordance to EN ISO 13919-1:1996 on a scale from 1 to 4. This ISO norm labels welds depending on their specific properties and their possible use. While the label



Beginning welding process Ongoing welding process Ending welding process

Figure 4.1: Raw images of the first weld of process 76.

4 indicates the ideal laser weld, denoted as B in the EN ISO 13919-1:1996, the label 1 implies a non-sufficient laser weld. The labels 2 and 3 mark the ISO classes D and C, respectively.

For learning a representative problem, the data has to ideally be evenly distributed, otherwise the result might be biased. Furthermore, the test set is supposed to be completely independent from the training data to make sure the evaluation of the algorithms capability for generalization is not compromised. For the available laser welding processes, the distribution of each class in the process can be seen in Table 4.1. As it can be seen, none of the processes has a completely evenly distribution. Of the four processes, the four different classes are best distributed in the process P77. In addition, this process contains the most images, resulting not only in the most evenly distributed, but also the biggest data set. Another issue arises from dependencies between the video data. When using video stream data with a high sample rate, like the laser welding data, one has to be aware that each image is not completely independent from the previous one, due to the short time interval of only 0.001s between the images. This fact implies that a training/test split can only occur in the weld level, as the interdependencies within one weld are to strong. Therefore the cross-validation will be performed by dividing the data set into different welds and assigning them to the training or test set respectively. While this split between training set and test set is not ideal in comparison to standardized data sets like the MNIST one, given this real data set the suggested split seems to be the most reasonable. A way to test the explanatory power of the trained machine learning approaches would be to also test them on a standard data set. As one of the challenges for the machine learning algorithms is whether they are robust against variance within the data, in a final step certain degrees of variance were re-introduces in a controlled way. Typical variance in laser welding consist of rotational, translational and scaling variance. The natural variance between different processes was measured and different amounts of variance from 0% variance up to 200% of the natural occurring variance was included in the data set.

Before feeding the images into the learning algorithm, they are further preprocessed. First, a region of interest (ROI) of the size 105 × 105 is applied. As the most information is contained in the keyhole and the heat bed, while the surrounding metal, filling out the rest of the image, is highly repetitive, this is a reasonable choice to decrease the amount of data that has to be processed without losing information. In a second step, the remaining part of the image is subsampled, resulting in a 32 × 32 images. It has been shown that this representation still contains enough information [62].

	Class 1	Class 2	Class 3	Class 4
Process 76	52,636	59,135	44,500	11,099
Process 77	83,493	36,085	49,772	22,421
Process 80	1,951	22,589	14,620	14,821
Process 81	6,343	33,827	10,859	11,608

Table 4.1: Label analysis of the complete data set in order to decide upon segmentation.

4.2 Intelligence-inspired control architecture

As explained in 2.1, a classical closed control loop is not always feasible for highly complex and individual processes. Due to variance in the sensor data, introduced by (small) changes in the sensor setup, e.g. translation, rotation or scaling of image data, a classic closed loop control approach will most certainly fail due to the incompatibility of sensor readings. Additional variance may be introduced due to changing environmental or material conditions. Furthermore, for a variety of modern industrial processes, the output of the system in terms of a reliable quality measure can not be provided. It is therefore not possible to close the loop, which is a condition for automatically learning and controlling such processes in a closed loop way.

To apply closed loop control to this kind of process, an intelligence-like architecture has been created to imitate the way humans approach complex problems, which is depicted in Figure 4.2. It is based on three principles, namely Representation, Prediction and Control. In a first step the incoming sensor data stream is converted into (invariant) information, similar to the information processing in the human visual cortex by using deep learning. These invariant process information, called features, contain all necessary information about the current process state and can then be used for further control steps. It further has been shown that, once learned, a neural network can process the necessary amount of computation within the real-time requirements of the laser welding process [178].

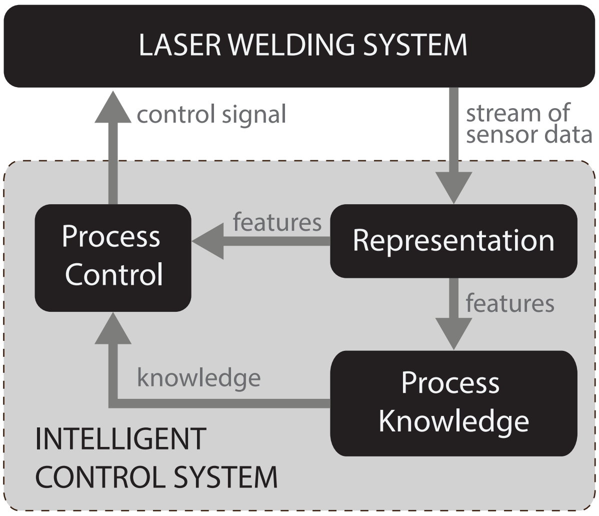


Figure 4.2: Intelligence inspired architecture [72].

In a second step, these features are then used to build up knowledge about the process from previous learned samples. By using general value functions to learn predictions from the extracted features, it is possible to make (temporal extended) predictions about the true quality of the process and therefore create a systems output for the feedback loop of a closed loop control approach.

When combined, the features and the predictions can later be used by a control approach, e.g. a policy gradient reinforcement learning algorithm, to find the correct control signal in order to apply the correct laser power to the system. These three building blocks together now closed the control loop for a complex industrial process that could not be controlled before.

They form a complete system that can represent its sensory inputs in an invariant way, build up knowledge about the process to make predictions and then evaluate the current and future performance in order to take the correct actions. This approach for an intelligence inspired control architecture can be seen in Figure 4.2 [72]. Each individual step will be further elaborated and evaluated in the following sections.

4.3 Feature extraction by deep learning

The ability to automatically extract the important information from data is a very important step towards autonomous artificial intelligence [179]. There exist a wide variety of established ideas for dimensionality reduction in order to transform raw sensor data into a low-dimensional and transformation-invariant representation of the system's state. Among all these techniques, deep learning [180] has demonstrated its capability to produce the lowest classification error for various problems when used for feature extraction [98].

Deep learning was inspired by the way the visual cortex of mammals works [125]. In the very first layer, the V1, simple edges and lines are detected, similar to Gabor filters [181]. These very basic features will be combined to geometric figures, like circles and rectangles within the next hierarchical step, the V2. The higher within the visual cortex the information is processed, the more abstract and complex the information will be represented - until it finally will be represented in a very robust and compressed way by just a “simple” word. This idea and structure was adapted in form of stacking neural network layers on top of each other in order to learn features from features and therefore gain more abstract representations[182]. Unfortunately while moving from a neural network with just one layer to a stacked neural network, the performance decreases due to the optimization function becoming non convex. In 2006, Geoffrey Hinton introduced the idea of Deep Learning to overcome this problem [183]. The parameters will not simply be adapted by backpropagation through the whole network, but first in a layer-wise pretraining. By keeping the number of hidden layers at the number of one during the training process, the training will more likely result in a better performance. After the layer-wise training is eventually done, the backpropagation algorithm is applied to the whole network again for a weight finetuning.

Ideally the approach used for the invariant perception has to be an unsupervised algorithm. This way it can consistently learn and improve its performance without human assistance. While neural networks are widely applied as classifiers in computer vision [91, 92], they can also be used in an unsupervised manner to extract features by using the autoencoder architecture [108]. Autoencoders have furthermore successfully competed with most state-of-the art feature extraction techniques (e.g., principal component analysis, linear discriminant analysis) [183] or improved [184] or directly learned [185, 145] complex (nonlinear) mappings for high-dimensional image data in combination with reinforcement learning. Autoencoders have demonstrated a remarkable capability of achieving more general representations, which can lead to to more robustness against varying data and overfitting [108]. Even better results are achieved by combining autoencoders with deep learning.

The idea of an autoencoder with one hidden layer has already been introduced in subsection 2.4.1. It will now be extended by stacking several layers on top of each other and generating a stacked autoencoder, as suggested by deep learning. The deep learning training procedure for a stacked autoencoder can be seen in Figure 4.3. For the first step of the training (a), an autoencoder with only one hidden layer is trained to generate the original input from a corrupted input, denoted by the neurons with an “x”. Once the training is complete (b) and the weights between the first and the second layer are fixed to a MLP. A second training step (c), including a new hidden layer which is stacked on top of the first one is performed. The inputs are corrupted with noise again and the neural network is

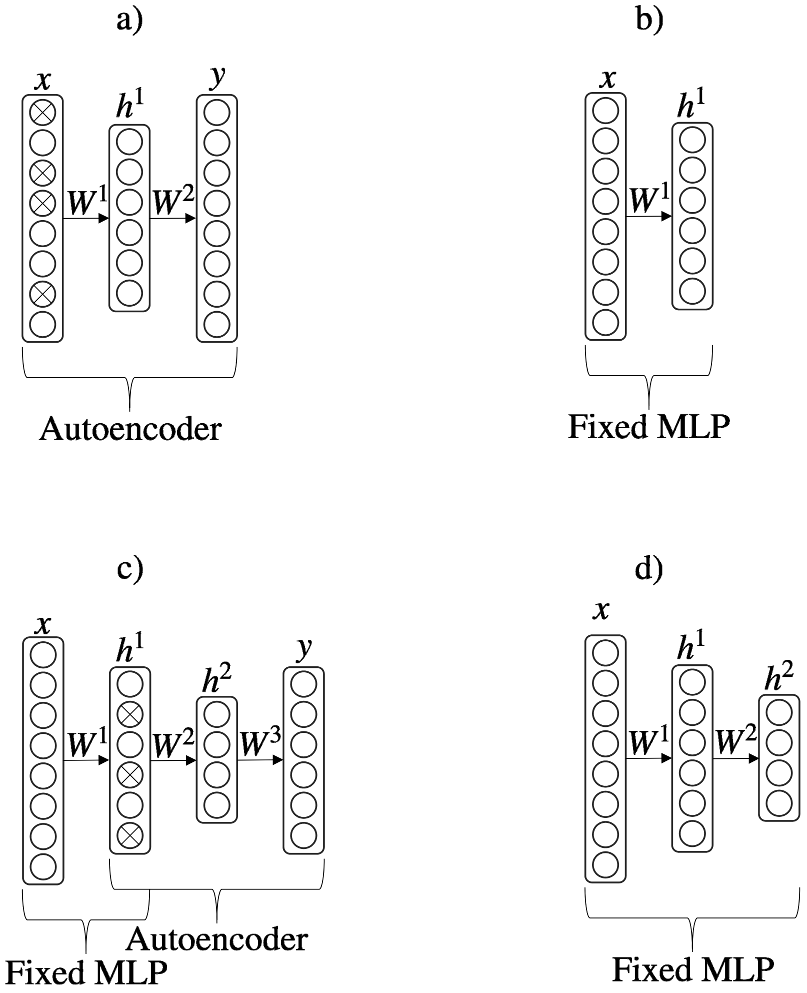


Figure 4.3: Deep learning approach for a stacked denoising autoencoder.

supposed to learn to recreate the uncorrupted. When the training procedure is complete (d), the learned weights for the new layer are fixed, creating a MLP with two hidden layers. The training procedure is continued until all layers have been trained in a layer-wise pre-training, followed by a backpropagation training for the whole architecture.

In this work, the performance of a stacked denoising autoencoder (SDAU) and a stacked convolutional autoencoder (SCAU) will be compared. The stacked denoising autoencoder and the stacked convolutional autoencoder follow the most common approaches of fully connected and convolutional neural networks and are therefore a reasonable choice for extracting invariant features in an unsupervised manner. To ensure comparability, both autoencoding approaches had the same constraints, regarding the time to find the final architecture, the parameters and the computational power.

4.3.1 Stacked denoising autoencoder

Stacked denoising autoencoders combine the already introduced principles of fully connected autoencoders and deep learning. They can be divided into two major groups. The first are so called triangular autoencoders and the second group are rectangular shaped. In a triangular autoencoder the layers decrease subsequently in size (number of neurons) until the bottleneck and the increase again afterwards [183]. For a rectangular SDAE the size of the hidden layers between the input layer and the bottleneck layer is kept constant [186]. For the specific application of laser welding images, the experiments yielded a better performance for rectangular shaped autoencoders. Different numbers of hidden layers within the range of $2, 3, 4, 5, 6, 7$ layers with an individual number of $1024, 2048, 4096$ neurons per layer were tested, resulting in a total of 28 different autoencoder configurations. In accordance with the suggestions in [108] the first hidden layer increases in size, compared to the input layer to allow a more general representation of the data. As sensor readings and camera images in special are usually corrupted with noise, during the training Gaussian white noise was applied, which is a common training method for stacked autoencoder [187]. The noise was adapted in each layer following the equation

$$x_{corrupted} = x + \mathcal{N}(0, [c * \sigma(x)]^2), \tag{4.1}$$

where x is the input signal for each layer, $\sigma(x)$ is the standard deviation of the input signal, c is a scaling factor and $\mathcal{N}$ is a gaussian distribution. The factor c is set to 0.3 to avoid unreasonable high activation due to the corruption. The corruption rate was 100%, as noise from sensor readings would also afflict the whole image. The final architecture consisted of six fully connected layers in the encoder and the decoder respectively. In the bottleneck the information was compressed to 16 features as this yielded the best results [67, 72]. The final architecture is depicted in Figure 4.4. The first layer (and also the following layers) increase in size to achieve a more general representation. There are 6 hidden layers in the encoder until the data is fed into the bottleneck. The decoder mirrors the encoders structure. Each hidden layer has 4096 neurons.

Parameter	Value
Batch size	128 images
Activation function	Rectified linear unit (ReLU) [118]
Linear decaying learning Rate	0.1 - 0.0001
Linear increasing momentum	0.9 - 0.999

Table 4.2: Common parameters for both autoencoder approaches.

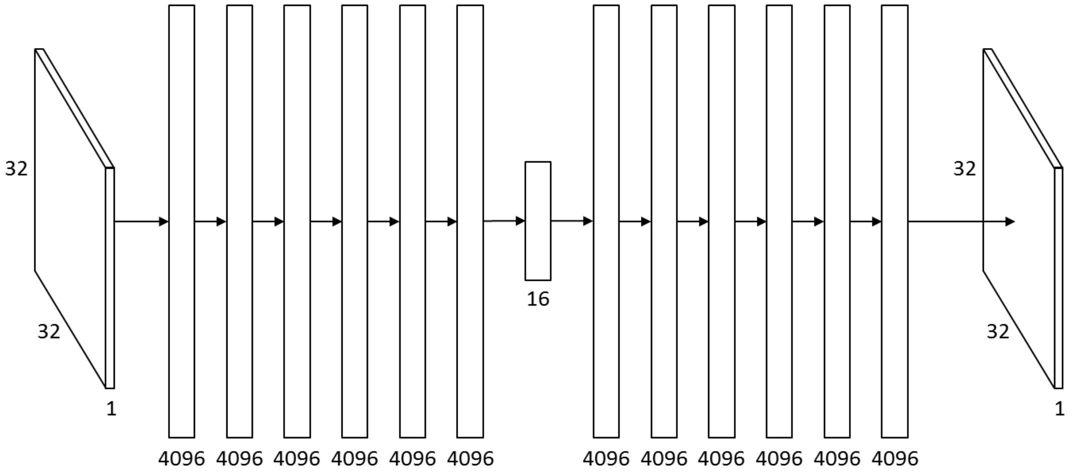


Figure 4.4: The final stacked denoising autoencoder architecture.

4.3.2 Stacked convolutional autoencoder

Stacked convolutional autoencoders employ both techniques from convolutional neural networks and autoencoders. So far, stacked convolutional autoencoders have not been widely applied to real-world problems. However, there exist a broad range of convolutional neural networks for image classification and recommendations on how to structure those networks, e.g. the “AlexNet” architecture [98]. For this thesis, several different combinations of convolutional and max pooling layers have been evaluated. An overview of the tested specific architectures can be found in Table 4.3. Additional to the architectural choices there are few parameters to decide upon in stacked convolutional autoencoders. For the convolutional layers, the number and size of filters have to be decided. As the input data has only a dimension of 32 × 32 pixel, the size of the filters were decided to be 3 × 3 with a stride of 1 and a padding of 1. While larger filters would consider to big amounts of the input image at a time, smaller ones would be to fine to extract edges. For the number of filters in a convolutional layer research indicates that it should increased by $150\% - 250\%$ for the first layers and after that be kept constant [98, 181, 188]. For the max pooling layers, three different pooling sizes—2 × 2, 3 × 3 and 4 × 4—were tested. Given the dimension of the initial data it was to be expected that the bigger the max pooling size, the more important information were to be lost. This was confirmed by the results and therefore only a single max pooling layer of the size 2 × 2 was applied in the final architecture. As suggested in [98], between the max pooling layer and the bottleneck layer, two fully connected layers of the size 4096 were added. Out of the tested fully connected layer sizes, i.e. $\{1024, 2048, 4096\}$, these yielded the best results. The final architecture is shown in Figure 4.5. The decoder is not depicted, but resembles the mirrored encoder. The complete final encoder consists of 5 convolutional layers with respectively $4, 8, 16, 16, 16$ filters, resulting in 60 filter of the size 3 × 3 in total. The convolutional layers are followed by a

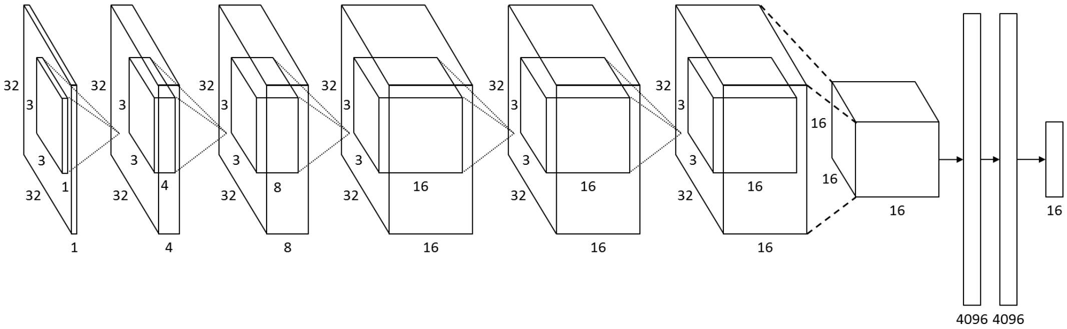


Figure 4.5: The final stacked convolutional autoencoder architecture from the input till the bottleneck.

	1 Pooling Layer	2 Pooling Layer	3 Pooling Layer
1 Convolutional Layer	CP	-	-
2 Convolutional Layer	CCP	CPCP	-
3 Convolutional Layer	CCCP	CCPCP	CPCPCP
4 Convolutional Layer	CCCCP	CCPCCP	-
5 Convolutional Layer	CCCCCP	CCCPCCP	-
6 Convolutional Layer	CCCCCCP	CCCPCCCP	CCPCCPCCP
7 Convolutional Layer	CCCCCCCP	-	-

Table 4.3: Different evaluated SCAE architectures.

max pooling layer of the size 2 × 2. Immediately before the bottleneck there are 2 fully connected layers with 4096 neurons each.

To give insights into the stacked convolutional autoencoder, the way the laser welding image is processed during its course through the stacked convolutional autoencoder is shown in Figure 4.6. The original image on the left is subsequently processed by the convolutional layers and then the max pooling layer. The fully connected layers and the bottleneck are left out in this visualization, as they can not be interpreted in a spatial structure, similar to the rest. While the images are fed forward through the network, the dissimilarities between the activation maps consistently increase. This can be interpreted as an increasing level of specialization, as the images proceed within the layers. The distinction between the representations is the highest in the layer, immediately before the bottleneck layer. This hints to these layers having a focus on certain distinct but different aspects of the image. Another interesting behaviour that can be seen on close observation regards the impact of the max pooling layer. As the max pooling has a size of 2 × 2, 75% of the input data is lost. The previous convolutional layers find a way to preserve the important information by shifting the activation maps by one pixel. The maps are therefore not aligned any more. That way, all information is stored in different activation maps and can be passed through the max pooling layer. A third interesting observation does regard the general activation

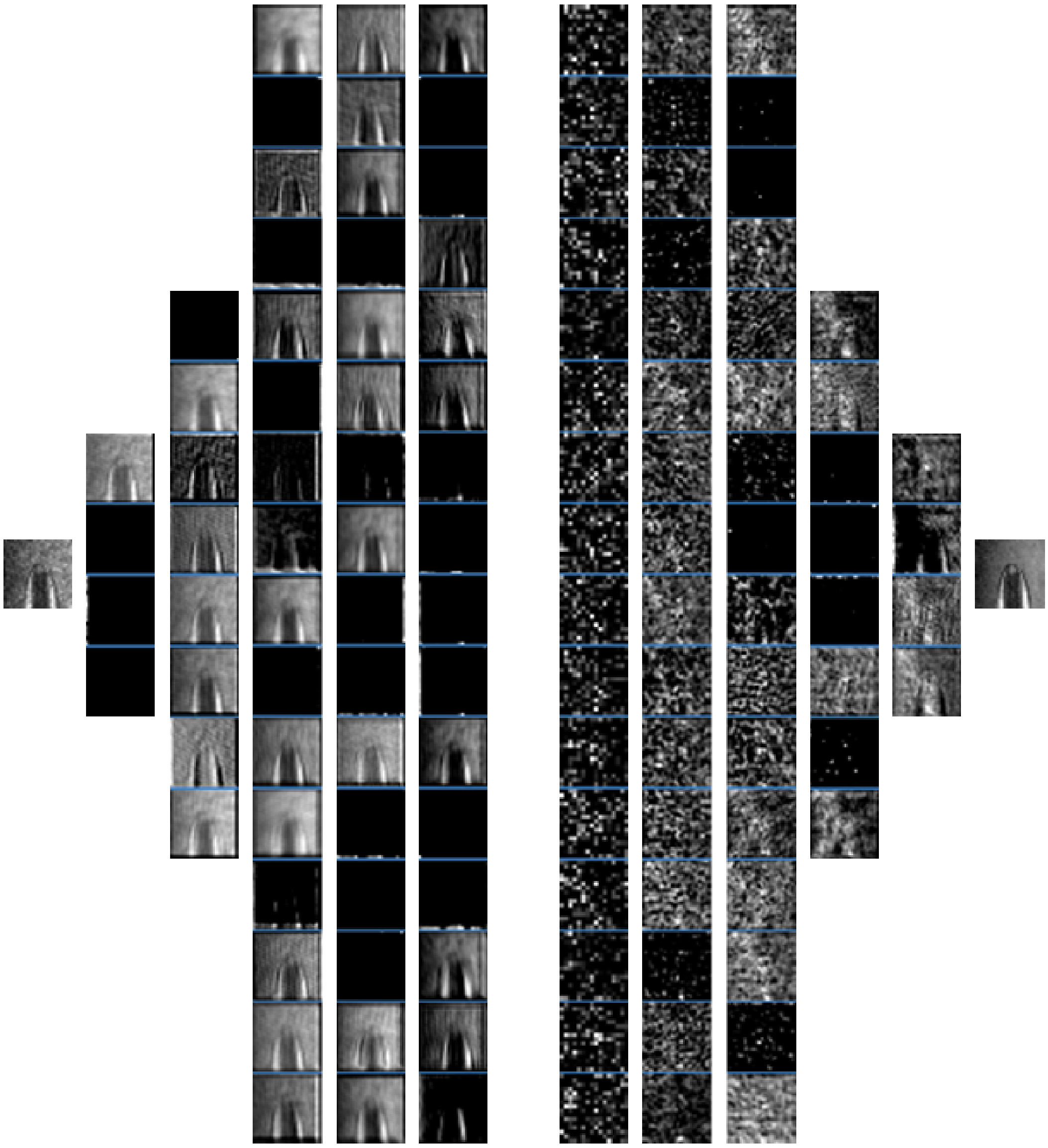


Figure 4.6: Visualization of a welding image, processed by the convolutional stacked autoencoder.

of the convolutional filters. While most of them resemble the input image with focus on different areas, some other seem to be mostly black on visual inspection. Further inspection of the corresponding weights and biases reveal that these maps are not inactive at all, despite appearing to be so. The weights indicate that these filters are highly activated by only a small amount of the input image. One could speculate that this behaviour leads to a focus on very distinct features of the input image, while most of the rest seems trivial to this particular filter.

4.3.3 Results

For the invariant perception, two different autoencoding procedures are tested and evaluated as well as an algorithm based on the raw image to provide a ground truth. The first autoencoder is the fully connected stacked denoising autoencoder (SDAE) [108] and the second is the stacked convolutional autoencoder (SCAE). To evaluate the performance of the autoencoder two different measures will be considered. The first measure will be the root mean squared error between the input image and the output image, which is also the optimization target, according to the equation 2.5. However, as not reconstruction, but using the extracted features for classification is the final goal, the reconstruction error might not yield the best error measure. Therefore the extracted features are further used to classify the image according to the introduced quality labels from EN ISO 13919-1:1996. This will result in a F-score that will provide a measure for the amount of captured information. The F-score will also be evaluated for artificial introduced variance to test the extracted features for robustness.

The most natural way to evaluate the performance of an autoencoder is to evaluate its reconstruction error. A perfectly functioning autoencoder that is capable of compressing all information within its bottleneck layer would have a zero reconstruction error. However, this is not to be expected in a real-world application. The eventually achieved reconstruction errors for both autoencoder approaches can be found in Table 4.4, while Figure 4.7 shows its development over the training. For both feature extraction methods, the reconstruction error shows a steady decline over the training time. It is noticeable, that reconstruction error for both approaches is very small. However, the stacked convolutional autoencoder performs slightly better in terms of reconstruction. This was to be expected, as convolutional neural networks so far outperform fully connected neural networks in almost every challenge.

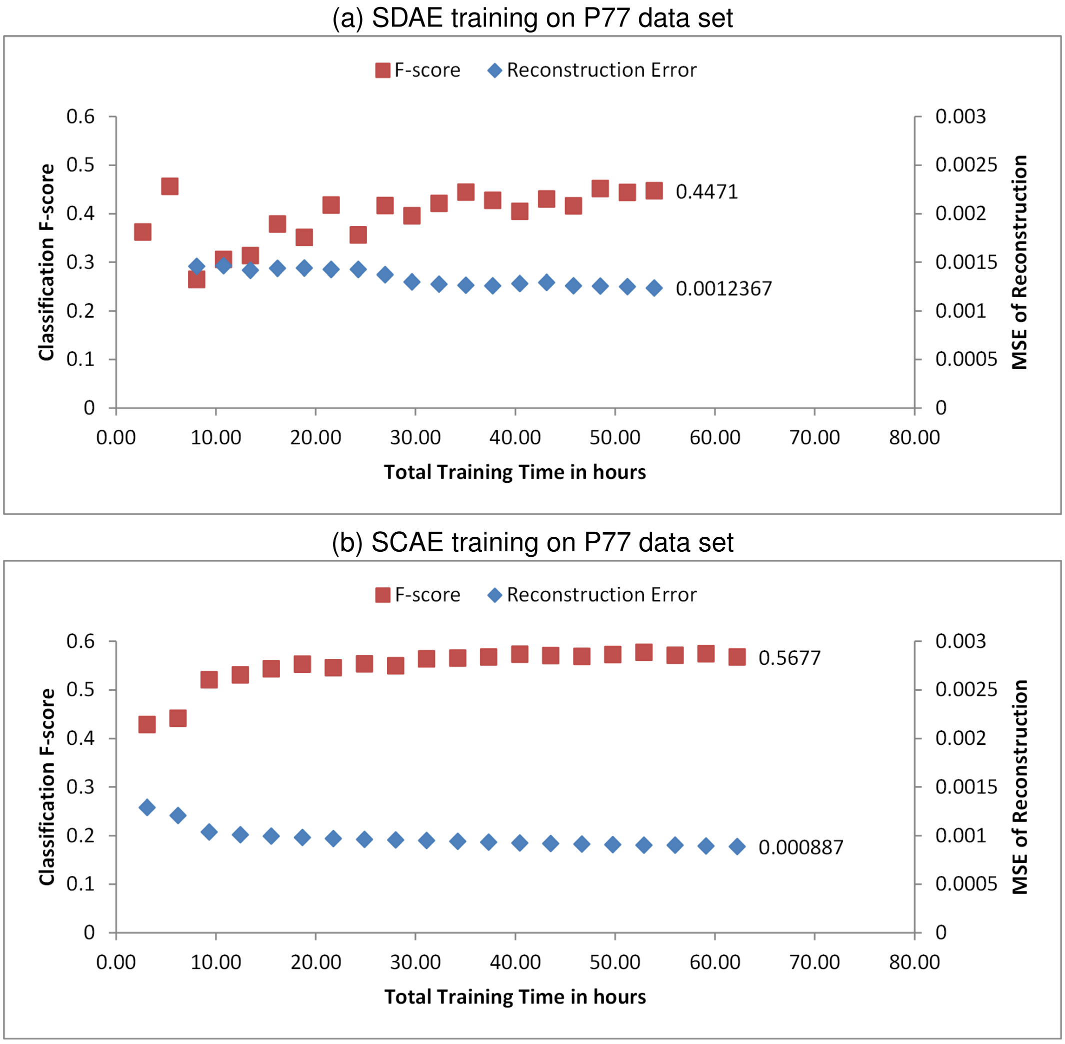
Although very intuitive and easy to understand, the reconstruction error might not yield the best measure in order to evaluate both approaches, given the amount of information within the bottleneck layer to be the important feature. Therefore, the trained architecture were used to extract their individual representation of the input image. These 16 features were fed into a support vector machine and then used to classify the input image as one of the true quality labels, given by EN ISO 13919-1:1996. Additionally, the uncompressed image with a resolution of 32×32 is fed into a SVM classifier to be compared. The results can be seen in Table 4.4.

A finding that might surprise at first is the fact that the pure image fed SVM demonstrates the highest F-score, questioning the whole approach of using autoencoders at all. There are several reasons for this behaviour. As the most information about the laser welding process can be found within the keyhole and the surrounding heat bed, the evaluation and classification might not be as hard a problem as it was thought initially. Furthermore, all the data in this process was recorded with a fixed, very precisely arranged set-up - minimizing or even eliminating any variance. And lastly, as there are interdependencies between the individual images, due to the high camera frame rate, the classification for these processes might become easier for a given classifier. For the two autoencoders the results are as expected. The F-score for the SVM, using the stacked convolutional autoencoder features is significantly higher than for the SVM, using stacked denoising autoencoder features. The features from the SCAE fed SVM do not only start at a higher *F*-score, while training proceeds, but also faster approach their maximum, resulting in a shorter training period, as it can be seen in Figure 4.7. Using features from the stacked denoising autoencoder, the F-score has its maximum after a very short training time to decline in performance afterwards. As the F-score never reaches this exceptional performance again, it can be seen as an outlier during the training period. The F-score for the SVM, using the stacked convolutional autoencoder features on the other hand demonstrate a steady improvement over the course of the training. This stable behaviour makes the training easier, as for this type of behaviour the training can be performed without human supervision and just be terminated after a certain time with the quasi-guarantee of achieving an adequate performance.

Data format	Best reconstruction error	SVM F-score
raw images	-	0.5895
SDAE features	0.0012367	0.4471
SCAE features	0.000886926	0.5677

Table 4.4: Reconstruction and classification performance for both autoencoding approaches.

While the experiments so far used a data set, in which the variance due to translation, rotation and scaling effects has been eliminated, the invariance towards this variance is of great importance. In a real-world laser welding application, certain degrees of variance will be introduced, due to the manual setup process. Therefore, artificial variance with specific amounts have been introduced to evaluate the approaches robustness against those. To decide the amount of variance, the natural variance between the four available processes has been investigated. This amount of variance was the chosen to be 100%. The artificially introduced variance was scaled from 0% up to 200%. Each type of variance has been tested individually and all three have been added together in a fourth test. Figure 4.8 shows the development of the F-score, dependent on the introduced variance. The F-score for the raw image fed SVM started with the highest score, as already explained, but drops very steep as soon as variance is introduced. Especially scaling and translational variance have an enormous impact on the raw image fed SVM. As these types of variance


(a) SDAE training on P77 data set

(b) SCAE training on P77 data set

Figure 4.7: Reconstruction and Classification performance for the final autoencoder architectures over the course of training.

do have a stronger impact on the keyhole, namely moving it or changing its size, the assumption that this part is mainly responsible for the classification performance of the raw image fed SVM is supported. Between the SDAE and the SCAE the initial advantage of the SCAE over the SDAE is getting lost as the gap between both approaches narrow. However, there is a clear hierarchy and the SCAE outperforms the SDAE in all test. For all approaches, translational variance seems to be the biggest problem, as the performance deteriorates the most.

In addition it is interesting to note that the classification performance drops below 25% when all three variance types are combined at a level of 200% of the natural occurring variance. At this point, the classification is as bad as random guessing. Further inspection reveals that this happens especially for the class 4, showing a recall value of only 0.07 (see Table 4.5). The same behaviour can be observed for the SVMs, using the raw input image and the stacked denoising autoencoder features. A possible explanation would be that the information to identify a class 4 laser weld is located in a very narrow region, e.g. the keyhole. Due to the combination of exceptionally high levels of scaling and translational variance, this information is shifted out of the algorithms focus, rendering the correct identification impossible.

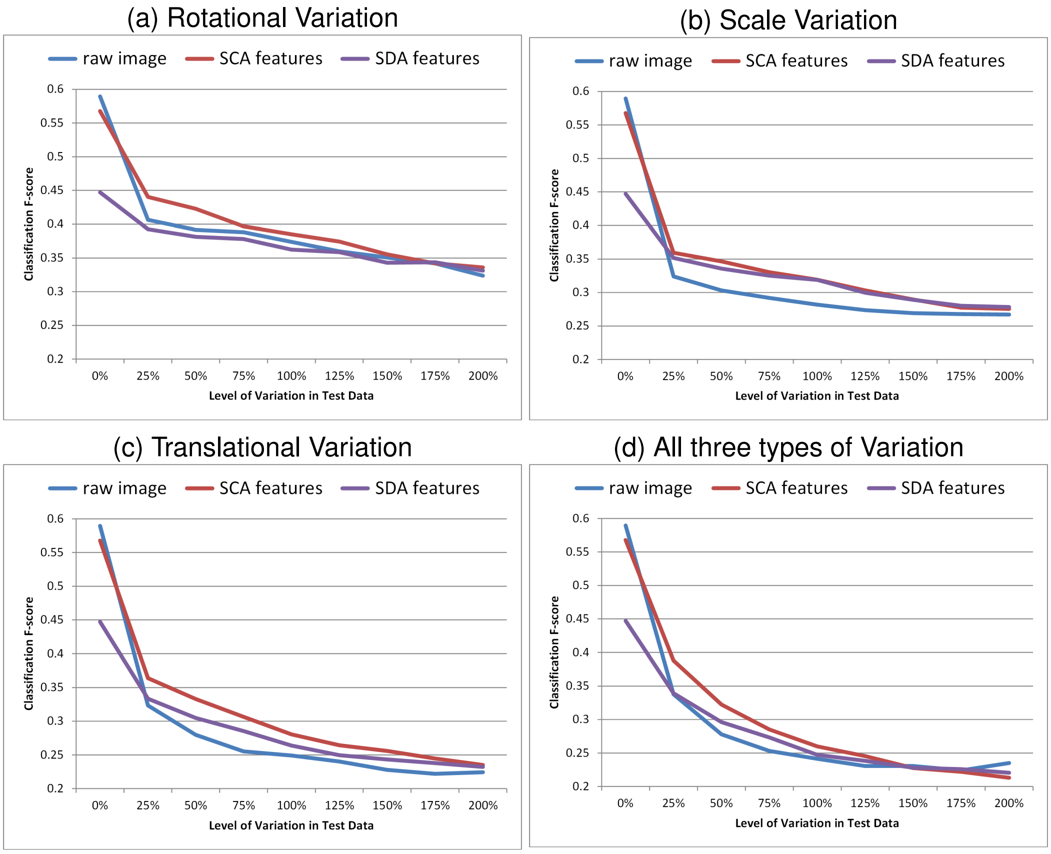
To further evaluate the final autoencoder structures, they were also tested on a well known and carefully composed data set, namely the MNIST. This is a standard data set that is ensured to have independent samples and is therefore widely used for testing autoencoding and classification algorithms. The tests were conducted to ensure the final results were not achieved due to the dependencies between the samples in the laser welding data set. The results can be found in Appendix B.

class	precision	recall	f-score	support
1	0.24	0.52	0.33	3200
2	0.23	0.14	0.17	3200
3	0.26	0.30	0.28	3200
4	0.29	0.04	0.07	3200
avg / total	0.26	0.25	0.21	12800

Table 4.5: Class report of the SVM, using stacked convolutional autoencoder features at a variance of 200%.

4.4 Creating a feedback signal via general value functions

The next step within the suggested architecture towards a closed loop control is the gathering and usage of knowledge. As there is no quality signal available in a laser welding process, this cannot be done by using a sensor, but has to be provided by an algorithmic approach. Similar to the knowledge of human beings, first knowledge has to be learned before it can be applied in form of intuition and predictions. This ability of making predictions is considered to be one of the most important aspects of intelligence by researchers and psychologists [189]. Humans do not only make conscious long time predictions about their future on which they base important decisions like education, but also numerous unconscious short-time predictions about their environment. A term for the latter is nexting [190]. The term defines a short time prediction about the environment. Humans use those


(a) Rotational Variation

(b) Scale Variation

(c) Translational Variation

(d) All three types of Variation

Figure 4.8: Classification performance, dependent on variance within the welding images.

for example to anticipate the movement of surrounding objects, the course of a conversation or to anticipate the behaviour of other humans or animals.

A way to make use of this concept is to utilize reinforcement learning and specifically general value functions [191]. It has been shown that this can be done for a short time horizon and on multiple timescales in order to make predictions about the relation between sensor values and specific signals of interest [192]. Following this approach process knowledge can be built up, using a general value function to represent the quality of the laser welding process, based on the invariant features from the invariant representation, according to Figure 4.2. As this approach is using function approximation in combination with general value functions, it is computationally cheap and can be utilized to make multiple short timescale predictions on different time scales. Predicting the quality of the laser welding process at the very moment can be interpreted as adding a new sensor.

This newly created signal can then be used in a closed loop way to adapt a control signal. However, the ability of predicting the future, i.e., mapping from current states to future quality, can take closed loop control even further. If the quality of the process can be successfully predicted, control signals that prevent changes in quality can be applied, instead of just compensating them afterwards. The process quality works as a cumulant signal [147] and will first be learned, in a way analogous to incremental supervised learning using least-mean squared updates during a training phase and later be predicted. As ground truth in the training phase serves the expert evaluated and labeled laser welding data set. The ground truth is the true quality and serves as a comparison measure in order to evaluate the algorithms performance. The quality labels, ranging from 1 to 4, in accordance to EN ISO 13919-1:1996 serve as cumulant signal, while the state will be provided by the autoencoder extracted features. To the 16 features the 3 photo diode values are added, resulting in a 19 dimensional state vector. The reinforcement learning algorithm will then use the tile coded representation of the state to calculate its predicted value, using the weight vector w. Together with the real value, provided by the cumulant, the predicted value $w^\top \chi(s)$ and the discounted prediction for the following state $\gamma w^\top \chi(s')$, the temporal-difference error δ is calculated, following line 4 in the Algorithm 3 in Table 4.6. In the next step, the actual learning occurs by adapting the weights w in order to reduce the td-error and move the prediction towards the td-fixpoint: $r + \gamma w^\top \chi(s') = w^\top \chi(s)$. During the learning process, these steps are performed for the whole training set, resulting in a weight vector w that becomes a compressed summary of the relationship between the sensory input $\chi(s)$ and the quality process (cumulant signal, r). Once the learning process has converged, i.e. all learnable information has been stored in the weight vector, it can be used for online prediction of the cumulant [193], which is the process quality. This creates a control signal for closed loop control.

4.4.1 Experimental details

Most sophisticated machine learning algorithms contain a number of meta-parameters that have to be decided upon. The nexting approach at hand has been evaluated, using cross-validation in order to find the most appropriate parameters among the tested one. The learning rate (step size) was chosen to be $\alpha = \frac{0.1}{m}$ out of the tested learning rates $\alpha \in \{\frac{0.1}{m}; \frac{0.2}{m}; \frac{0.4}{m}; \frac{0.8}{m}; \frac{1}{m}\}$, where m is the number of active tiles in the tile-coded features representation $\chi(s)$. The trace decay parameter was set to $\lambda = 0.6$ with $\lambda \in \{0.6; 0.8; 0.9; 0.95; 0.995\}$. Another parameter that is closely related to the tile coding itself is the memory size. It has to be chosen big enough in order to contain all the information, but small enough to make generalization possible. This is the classical problem between overfitting and generalization. Four different memory size were tested and it was eventually chosen to be $n = 1,000,001$, with $n \in \{1,000,001; 750,001; 500,001; 250,001; 100,001\}$, with n being the dimension of the weight vector.

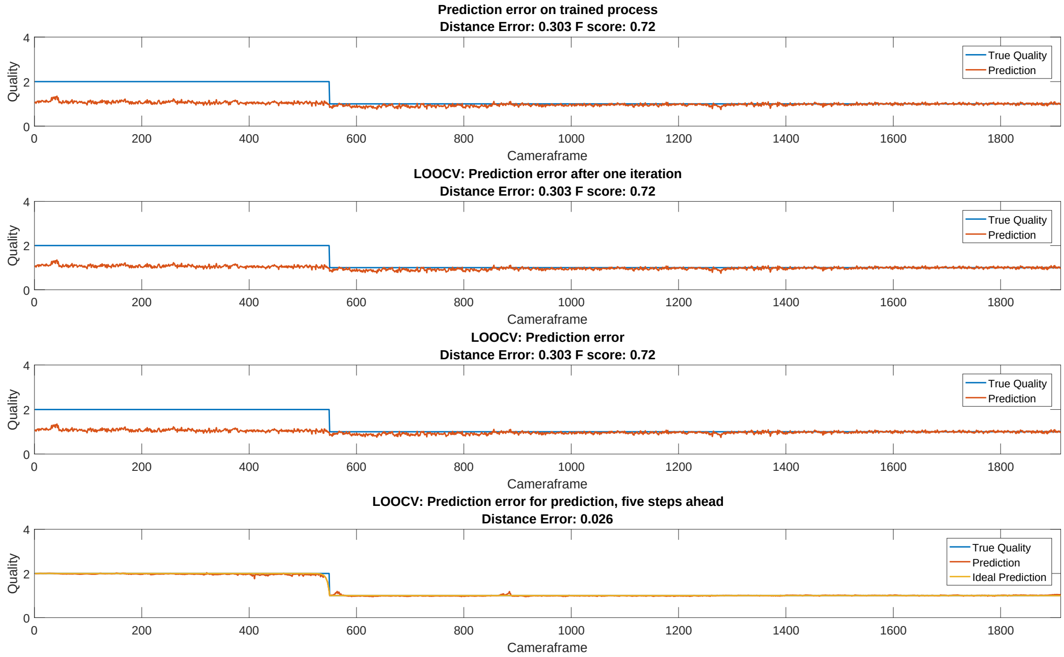


Figure 4.9: Nexting quality prediction, using stacked denoising autoencoder features.

Algorithm 3 Nexting with Temporal-Difference Learning

```
initialize: $w, e_w, s$
repeat:
    observe $r, s'$
    $\delta \leftarrow r + \gamma w^\top \chi(s') - w^\top \chi(s)$   // calculate the td-error, based on the current weights
    $e_w \leftarrow \gamma\lambda e_w + \chi(s)$   // update eligibility trace $e_w$, based on decay and visited state
    $w \leftarrow w + \alpha\delta e_w$   // update weight vector $w$, based on td-error
    $s \leftarrow s'$   // make successor state the current state
```

$w \in \mathbb{R}^n$:	value function weight vector	$e_w \in \mathbb{R}^n$:	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
$\chi(s) \in \mathbb{B}^n$:	tile coded state vector	$\alpha \in (0, 2)$:	learning rate
$\lambda \in [0, 1]$:	eligibility trace decay factor		

Table 4.6: Nexting algorithm for use in laser welding process quality prediction.

In the experiments, predictions on two time scales—the first one being the immediate quality and the second one being the quality five time steps in advance—were conducted and evaluated. The predictions of the current quality corresponds to $\gamma = 0$. This results in a slightly different equation for the td-fixpoint: $r = w^\top \chi(s)$, which is the mathematical expression for the prediction being the immediate quality signal. For evaluating the performance of this approach, two error measures are introduced. The first is the mean absolute error, or *distance error.* It provides a measure for how far the algorithms prediction is off in terms of the absolute difference between the prediction and the true quality. A second error for the case of $\gamma = 0$ is the *classification error* in terms of the F score. It is calculated by rounding the prediction to the next integer in order to match it to the quality classes, introduced by the EN ISO 13919-1:1996. It has to be noted that the classification error only makes sense for the prediction of the immediate quality as this classification task does not apply for a temporal extended prediction, i.e. $\gamma > 0$. The performance of the nexting approach for $\gamma = 0$ can be seen in Figure 4.9 for using stacked denoising autoencoder features and in Figure 4.10, using the stacked convolutional autoencoder features, respectively. The chosen example was a weld with a changing quality during the process, as this turns out to be the most challenging prediction task, compared to a constant quality prediction. The experiment was an overlap weld with a constant welding velocity of 3.5m/min and a constant laser power of 2000W. The welding seam was intentionally contaminated with grease, as it might happen due to negligence in a production environment. As the laser hits the grease at frame 550, the grease starts to burn, resulting in a drop of quality and an insufficient weld. The weld stays insufficient for the rest of the weld.

4.4.2 Results

The **first subplot** demonstrates the algorithms capability to learn the problem. The corresponding weld was the only member of the training set and test set. While this experiment does not offer any information about the generalisation ability of the nexting approach it is an indicator whether the features contain enough information about the problem to learn the dependency between the sensor data and the true quality at all. This is the first requirement in order to later learn from the whole data set and correctly predict unknown welds. The features of the stacked denoising autoencoder do not provide sufficient information, as the predicted quality by the nexting algorithm is not able to follow the true quality. The stacked convolutional autoencoder features on the other hand result in a perfect tracking of the true quality, when used as inputs.

The **second subplot** in both figures shows the testing performance after just one iteration of leave one out cross validation, in which weld 99 was left out of the training set. The algorithm has learned from just one iteration the very basic dependency between the provided features and the quality of the laser weld. The distance error for the test, using stacked denoising autoencoder features is 0.303, while the distance error for using stacked convolutional autoencoder features is 0.299. It can be seen that the stacked convolutional autoencoder features provide a better representation as the generalisation from the already seen welds to this unknown weld is better in terms of the error.

The **third subplot** shows the performance of a LOOCV for weld 99 after the learning has converged. In Figure 4.10 the nexting algorithm fails to detect the acceptable welding quality from frame 1 to 550, where the grease is located, based on the stacked denoising autoencoder features. This performance results in a distance error of 0.303 and a F score of 0.7. The generalization, provided by these features seems to be limited for the weld 99, although the general LOOCV distance error of the whole cross validation is comparably low, being 0.019 and a F score of 0.924, as can be seen in Table 4.7. The prediction, based on the stacked convolutional autoencoder on the other hand are performing better in general, achieving a LOOCV distance error of 0.015 and F score of 0.928, in special on weld 99 with a distance error of 0.015 and an F score of 1.

In the **last subplot**, the prediction for γ = 0.8 can be seen. This corresponds to a prediction 5 timesteps into the future. It is interesting to notice that while this task can be considered more challenging than predicting the immediate reward, the prediction performance, using stacked denoising autoencoder features, actually improves to a distance error of 0.026. For the LOOCV the error is the same as for predicting the immediate quality. The performance on weld 99 with the stacked convolutional autoencoder features decreases slightly to 0.021, but stays better than for the competing approach. The LOOCV error increases slightly to 0.018.

Summarizing the experiments, the nexting algorithm is capable of predicting the correct quality with a very high precision for the immediate reward (γ = 0) and for the future (γ = 0.8). The features, provided by the stacked convolutional autoencoder demonstrate to be slightly more informative than the ones, extracted by the stacked denoising autoencoder.

Experiment	SDAE features	SCAE features
LOOCV distance error with γ = 0	0.019	0.015
LOOCV F score with γ = 0	0.924	0.928
LOOCV distance error with γ = 0.8	0.019	0.018

Table 4.7: Error measures for Nexting, using different features.

4.5 Control approach using reinforcement learning

The last step to enable a closed loop control approach is to include a decision taking algorithm in order to create the actual control signal. Given the correct feedback signal there exist a wide range of control approaches. However, most of them have the drawback of requiring extensive human setup, mathematical models or are prone to unexpected disturbances. As the suggested architecture is supposed to provide a high-quality welding seam on its own, the controller has the be able to learn from experience and improve its own performance. A machine learning algorithm that has shown promising results for this type of tasks is actor-critic reinforcement learning (ACRL) algorithm [194], which will be

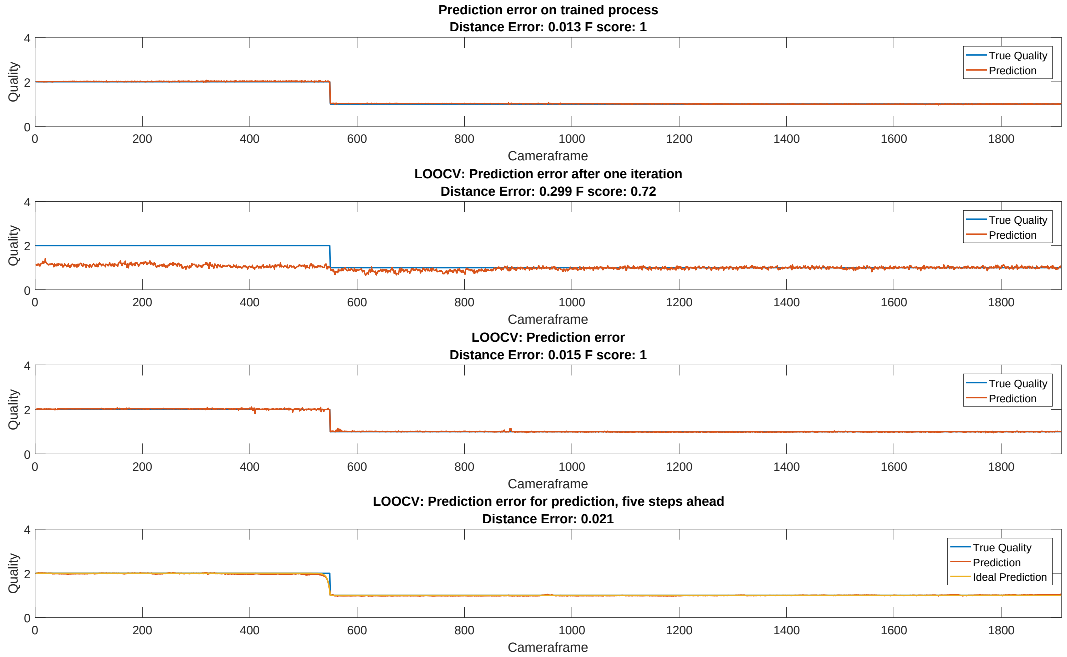


Figure 4.10: Nexting quality prediction, using stacked convolutional autoencoder features.

used to generate the control signal within the architecture. An ACRL consists of two parts, which are the actor and the critic. While the actor takes actions according to a learned policy, the critic consistently evaluates these actions. Several characteristics make it especially useful for the control problem at hand. Actor-critic reinforcement learning algorithms are parameter based, so experience the algorithm has already learned from does not need to be stored, which indicates a constant memory requirement over time [195]. They can be updated within milliseconds and the computation can be done in a linear incremental way. If combined with function approximation, actor-critic reinforcement learning algorithms scale very well to real-world problems, as it has been shown for different applications [196, 197, 198, 143].

4.5.1 Control simulation details

The experiments with the controller were not possible to be conducted, using the real laser welding station, due to time and cost issues. Therefore, a laser welding simulation, based on [199] was programmed. It provides the welding seam depth based on the welding seam width, with a welding depth ranging from 10mm to 20mm and a welding seam width of 1.2mm to 4.4mm. The possible applied laser power ranged from 1.5kW to 5kW. Although being nonlinear, this preliminary welding simulator is far from being as complex and challenging as a real laser welding process. Following the suggestions of industrial laser welding experts, several changes and limitations were introduced into the simulation to design it more closely to a real laser welding process: To simulate noisy sensor readings, white Gaussian noise with the standard deviation $\sigma = 0.05$ was applied to the welding seam width, corresponding to the systems state.

Similar to real laser welding systems, the power cannot be changed by an arbitrary amount within a short time horizon. The simulation was therefore restricted to adjusting the power by 200W per control iteration, regardless of what laser power the control algorithm requested. In a real laser welding process, a certain amount of the input energy is absorbed by the surrounding metal, while welding. This will result in a slower physical response to changes in the laser power. It was therefore reasonable to restrict the welding simulator in a way that the welding width can only change by 0.3mm per iteration. This corresponds to 10% of the complete welding seam width range. Together, these limitations and restrictions form a more accurate and challenging simulation for a laser welding process in order to evaluate the capability of an actor-critic reinforcement learning algorithm to apply the correct control signals.

The detailed code for the complete algorithms, including the decision making in the actor, the evaluation step in the critic and the updates for all variables, can be found in Table 4.8. The notation is the same as in the nexting algorithm in Table 4.6.

The control algorithm has only access to the laser welding systems laser power. As the power is a continuous action space, a continuous-action actor–critic reinforcement learning (ACRL), with a one dimensional action space is chosen, as outlined in [195, 143]. While the system only allows laser powers within the range of $[1.5, 5]$, the actor-critic does not know about this constraint. The actions were mapped and clipped to the systems power range to ensure only allowed laser power adjustments outside the actor-critic and without feedback to the algorithm. It samples its actions from a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with mean μ and standard deviation σ which results in a stochastic control policy $\pi(a|s)$. While the idea of using a stochastic policy for a control problem might be counter-intuitive at first, it turns out to be a reliable approach, as the mean μ and standard deviation σ are linear combinations of learned and adapted weight vector and the feature vector as shown in Table 4.8, lines 3 and 4. Therefore the critic can adapt both parameters to adjust mean and standard deviation to result in appropriate control signals. The standard deviation can be interpreted as a measure for curiosity of the algorithm or in other words its likelihood to explore. A large standard deviation will result in a very broad Gaussian distribution and therefore a large number of possible chosen actions. Exploratory steps usually result in more knowledge about the problem and are therefore necessary to find the appropriate actions. On the other hand they will usually result in a poor long term performance, when still chosen although the best action is already known. A good balance between exploration and exploitation is to explore in the beginning and then consequently decrease the exploration in favor of exploiting the gained knowledge. This is achieved by the critic evaluating the performance and adjusting the standard deviation to decrease, as soon as the

Algorithm 4 Continuous-Action Actor-Critic Reinforcement Learning [195]

```
1: **initialize:** $w_\mu, w_\sigma, v, e_\mu, e_\sigma, e_v, s$
2: **repeat:**
3:     $\mu \leftarrow w_\mu^\top \chi(s)$                              // compute the current mean for the Gaussian distribution
4:     $\sigma \leftarrow exp[w_\sigma^\top \chi(s) + log(\sigma_c)]$      // compute the current std for the Gaussian distribution
5:     $a \leftarrow \mathcal{N}(\mu, \sigma^2)$                        // randomly choose action from Gaussian distribution
6:     **take action** $a$, **observe** $r, s'$
7:     $\delta \leftarrow r + \gamma v^\top \chi(s') - v^\top \chi(s)$       // calculate the error, based on the current weights
8:     $e_v \leftarrow \lambda_c e_v + \chi(s)$                         // update the eligibility trace $e_v$, based on decay and visited state
9:     $v \leftarrow v + \alpha_v \delta e_v$                           // update the value function $v$, based on learning rate and error
10:    $e_\mu \leftarrow \lambda_a e_\mu + (a - \mu)\chi(s)$             // update the eligibility trace $e_\mu$, based on decay and visited state
11:    $w_\mu \leftarrow w_\mu + \alpha_w \delta e_\mu$                  // update the weight vector $w_\mu$ based on the td-error
12:    $e_\sigma \leftarrow \lambda_a e_\sigma + [(a - \mu)^2/\sigma^2 - 1]\chi(s)$   // update the eligibility trace $e_\sigma$, based on decay and visited state
13:    $w_\sigma \leftarrow w_\sigma + \alpha_w \delta e_\sigma$          // update the weight vector $w_\sigma$ based on the td-error
14:    $s \leftarrow s'$                                               // 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
$\chi(s) \in \mathbb{B}^n$:	tile coded state vector	$\alpha_v, \alpha_w \in (0, 2)$:	learning rates
$\lambda_a, \lambda_c \in [0, 1]$:	eligibility trace decay factors		

Table 4.8: Actor-critic algorithm with continuous-valued output actions for controlling laser welding power.

mean shifts towards the correct action. The result will be a narrow Gaussian distribution (at some point only a single action will be chosen, as sigma approaches zero) and a very steady performance. The performance is provided by a reward signal. This signal would be provided by the nexting algorithm in the fully integrated system as shown in Figure 4.2.

In the following experiments, the reward signal is based on the absolute difference between the requested welding depth and the actual welding depth. The reward signal has to be bounded as the standard deviation is calculated, using an exponential function, see 4.8 line 4. An unbounded reward would result in large td-errors, which can potentially lead to divergence. The reward is therefore calculated as $r = -0.5 + 1/[1 + exp(|d^* - d|)]$, where

d is the actual welding depth and d^* is the desired welding depth. This function results in a reward $r \in [-0.5, 0]$, in which the reward of 0 is only given, when the desired welding depth is achieved.

Similar to the nexting algorithm and the standard approach in reinforcement learning, each learning iteration consists of a quadruple (s, a, r, s') to be evaluated. The algorithm starts in a state s, provided by the features, extracted from the sensor data or the welding seam width in the simulator, chooses an action a from the set of actions, receives a reward r for the transition into the successor state s'. The td-error is calculated and the weights and traces are updated, resulting in an updated value function v and new parameters μ and σ for the Gaussian distribution. In the experiments, the following learning parameters were used: 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. The parameters were chosen by applying a grid search over the parameter space. For the learning rates, a fully factorial experimental design with the possible choices of $\alpha_v, \alpha_w \in \{\frac{0.0.05}{m}; \frac{0.1}{m}; \frac{0.2}{m}; \frac{0.4}{m}; \frac{0.8}{m}\}$. The trace decay parameters were chosen from the following set: $\lambda_a, \lambda_c \in \{0; 0.2; 0.4; 0.6; 0.7; 0.8; 0.9; 0.925; 0.95; 0.975; 0.99\}$. $\sigma_c = 1$ was chosen to be 1 to encourage the algorithm to explore in the first step, but does bias the standard deviation in further steps, due to $log(\sigma_c) = 0$. The trace and actor weight parameters w were initialized to zero, as no prior knowledge was available or was wanted to be included beforehand. This also results in a optimistic initialization, as the weight vector for the value function v was initialized at the ideal value. An optimistic initialization will cause the td-error around the correct welding depth to be small, which will result in only small updates and therefore a faster convergence time. As an optimistic initialization does not require knowledge about the process, this was a reasonable decision.

4.5.2 Results

The experiments were conducted for 30 independent runs to make sure the results are reliable and do not depend on specific initialization. The results were averaged and illustrated in Figure 4.11. The two topmost plots (a), (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. All of the trials converged towards the true solution despite the introduced challenges, which indicates a very robust approach. In the first line of Figure 4.11 the mean and the standard deviation for the Gaussian distribution can be seen. Both values behave as expected for a learning system, i.e. the mean shifts towards the correct value, while the standard deviation consequently drops to narrow the Gaussian distribution in order to sample around the correct action. The second line shows the applied laser power and the gained reward. The laser power in the beginning exhibits the expected distribution. As soon as the learning starts, the range of the applied power narrows down, until the algorithm eventually samples within a very close

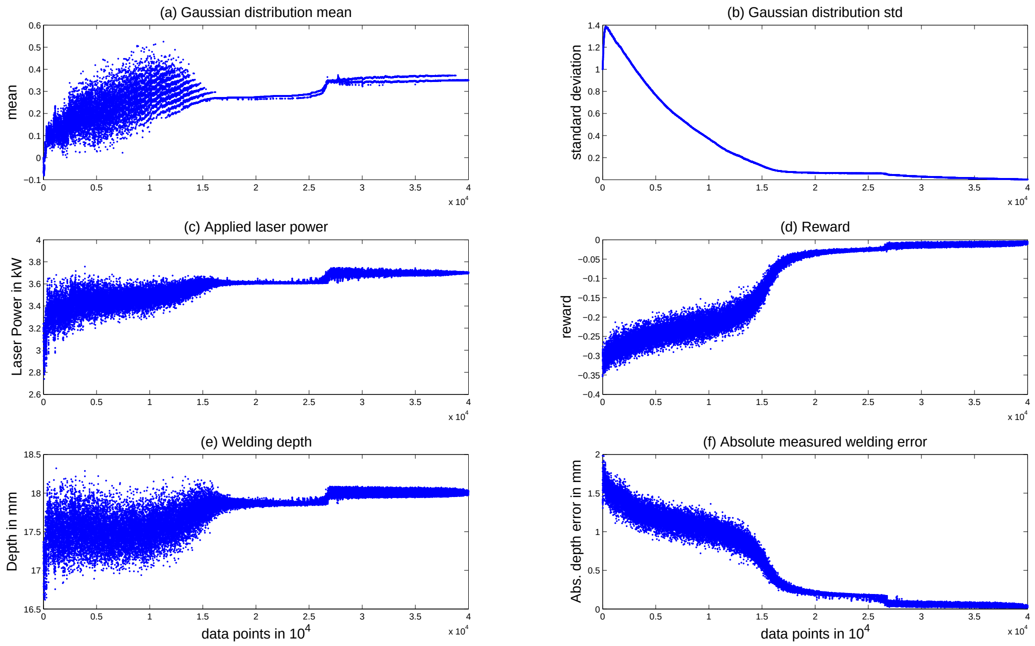


Figure 4.11: Average learning performance for the continuous actor-critic algorithm over 30 independent runs [72].

range around the correct solution. The gained reward constantly improves over the time, indicating the improving performance. The last line depicts the achieved welding depth and the error the algorithm makes. While the welding depth stabilizes at the desired depth of 18mm, the error converges towards zero. Eventually, the precision of the algorithm is only dependent on the amount of noise. Although the learner had to cope with a noisy and nontrivial system it reliably found the correct solution for the control problem and converged towards it in all trials.

From a computational point of view the algorithm performed fast enough for real-time requirements, which corresponded to 1ms. Each control and learning iteration for the actor-critic reinforcement learning algorithm took 0.34 ms. The computer for the experiment used an Intel Core i5-2400 with a 3.1GHz clock rate, 6MB of shared L3 cache, 4GB DDR3 RAM, and ran 64-bit Windows 7. The implementation was done in MATLAB, indicating additional potential for faster computation, using an appropriate compiler language, e.g. C++. The first publication of this experiment can be found in [72].

4.6 Conclusions and contributions of this chapter

In this chapter we introduced a new approach for industrial laser welding, based on a unique combination of machine learning algorithms. For this purpose we created a new laser welding data set. In this data set, the most common problems in industry were included, such as different pollution of the welding seam and changing laser speed. This data set can be used for further research in the field of applied machine learning for production engineering processes.

For the open loop laser welding system, we introduced a new combination that resembled important aspects of intelligence to provide an invariant perception, knowledge based predictions as a feedback signal and reinforcement learning based decision making for control signals.

The proposed combination of deep learning and general value functions is the first combination of these approaches. Due to the generic capability of these algorithms, they can easily be transferred to any open loop process, promising the create a reliable feedback signal to transform them into closed loop processes. The complete architecture is furthermore able to learn independently from human supervision. This ability is a step towards autonomously learning industrial processes and yields great potential for production engineering processes.

5 Discussion

In this work, two different data-driven approaches to improve or enable closed loop control have been shown. The first approach was to extend simple PID controllers by machine learning, general dynamic neural networks in particular. This approach is suitable for control tasks that follow the typical closed loop setup, i.e. a set point and system feedback are available. The approach can be seen as a stochastic adaptive control approach, yielding a completely automated adaptive control algorithm that does not rely on models which need to be provided by humans. It is therefore well suited not only for systems where disturbances can occur but also for systems that have changing system dynamics over time. The second part introduces an intelligence-like architecture that employs and combines several machine learning algorithms, including a combination of general value functions and deep learning. This approach is suited to complex control systems that cannot be sufficiently modeled and that may not have system feedback for closed loop control available. Both approaches and the experimental results evaluating those approaches will be individually discussed in the remainder of this section.

Machine learning extended PID control: This chapter introduced a way of using general dynamic neural networks to overcome the inherent limitations of PID controllers. As PID controllers only have three parameters, which are usually fixed before handing the control over, they do not perform well, except for in a very distinct working range. To enhance their performance on complex and nonlinear systems, the general dynamic neural network was allowed access to the PID parameters in order to adapt them during the running process. The approach allows PID control to be applied successfully to a wide range of control problems, suggesting a significantly more precise and robust performance.

The suggested adaptive PID controller was implemented and tested on four different systems that present the most common and challenging problems in control theory, including unstable and chaotic systems. Additionally, for each system, a standard PID controller and an additional traditional model-based control approach were implemented as benchmarks. All tests were run in four different configurations, including noise and disturbances to evaluate robustness against noisy measurements and unforeseen influences on the processes. The noise was Gaussian noise with a signal to noise ratio of $SNR = 20$. The disturbances were gradually increased in strength, until all but one controller failed to compensate. This resulted in a total of sixteen different experimental set ups that were controlled by three controllers each.

For all the conducted experiments, the adaptive PID controller demonstrated superior performance in comparison to the standard PID controller. Given the nature of the control problems and the limited capabilities of a standard PID controller this result was to be expected. While the standard PID controller was able to control all presented problems, its performance in terms of the root mean square error (RMSE) was worse than the performance of the adaptive PID controller. Especially, when presented with the disturbances, the standard PID controller had significant problems with each control task, completely failing to stabilize the inverted pendulum on a cart and the chaotic fluid system. Although the adaptive PID controller changed the PID parameters only slightly to adapt to the changing situations, is demonstrated a remarkable increase in performance. Additionally the controller output was lower in comparison, sparing the actuators and consuming less energy. The adaptive PID controller can be considered to be the superior control approach in terms of performance, when directly compared to the standard PID controller.

Compared to standard, model-based adaptive control approaches, the adaptive PID controller demonstrated superior behaviour in 13 out of 16 test scenarios. The smith predictor achieved a better RMSE for the LTI system with a non-neglectable time delay in both disturbed cases. This outcome can be expected, as the smith prediction has additional knowledge about the exact time delay, dramatically reducing the complexity of the control task. For a real system the time delay can mostly not be assumed to be known. In the experiments utilizing the thermal convection loop, the backstepping performs slightly ($8.9 * 10^{-1}$ to $9.0 * 10^{-1}$) better than the adaptive PID controller for the experiment with noise but without disturbance. If the disturbance is applied, the backstepping fails to control the system, entering a meta-stable state. Similarly, the LQ regulator for the inverted pendulum on a cart fails to compensate for the disturbance, while the adaptive PID controller manages to stabilize the system. Although the model-based approaches have access to the complete knowledge of the test scenarios, in terms of the differential equations, the data-driven adaptive PID controller manages to outperform them apart from the already mentioned three cases. It achieves more stable and better qualitative results.

In an additional experiment, the influence of the arbitrary connections is evaluated, comparing the control performance with one simple feed-forward neural network and on neural network with an external dynamic on the inverted pendulum on a cart. Not only does the adaptive PID controller have a significant higher convergence rate ($\approx$ 5.8 times) than the neural network with an external dynamic, the converged solution also provides a superior control performance ($\approx$ 1.8 times). The distance in terms of convergence and performance to the simple feed-forward neural network is even higher. It can be concluded that the temporal information that is encoded in the recurrent connections yields a significant improvement for closed loop process control.

One drawback of the suggested approach is still the implementation effort for the neural networks and the missing guarantee for convergence. Although the neural network learns to minimize the error between the set point and the actual systems output, it can so far not be guaranteed to converge to a stable and steady performance. However, for simple feed-forward networks, such a guarantee was already formulated [200] so a similar analysis will most certainly be provided for more complex neural networks in the near future. Furthermore, finding the ideal network structure for the general dynamic neural network and implementing it still requires efforts. One way to automatize this step would be to apply genetic or evolutionary algorithms that search for an ideal network structure autonomously [201, 202]. Unfortunately, it can not be verified that the neural net structure, eventually chosen, is the best one for the problem at hand, but the conducted experiments indicate only small performance fluctuations between the tested neural network structures. This can be seen as an inherent flaw to the usage of neural networks. However, the better control performance should outweigh the additional time and work.

As the neural networks only consist of a small number of neurons, the computation is not expensive. Calculating a pass through such a network does only require one matrix-vector multiplication for each layer i.e. three for the networks, used in this thesis. For the learning, the computations would double due to the backwards pass through the network. This computation can easily be done within a short time, using a low cost small single-board computer, like a Raspberry Pi.

New control architecture for production engineering: In this part, a new and unique combination of deep learning and reinforcement learning has been shown to enable closed loop control for processes without a feedback signal. The goal was to establish a closed loop control for industrial laser welding in particular. This process can so far not be modeled adequately by a mathematical model and in addition, the process quality can so far not be measured by a sensor. Given the nature of the process and the various number of changing environmental and material conditions, even a thorough in advance setup cannot guarantee a steady performance. Therefore each laser weld has to be inspected manually after the process, which is time and labour consuming. The suggested approach takes care of these issues by establishing an intelligence-like architecture that is inspired by the cognitive process of human beings. As described in Section 4.2 it consists of three parts: representation via deep neural networks, prediction via general value functions and control via an actor-critic reinforcement learning algorithm. Such an integrated architecture is well suited for a challenging and process with inherent changing conditions. While in this study only the application of laser welding was considered, the framework itself is flexible enough to be applied to other similar industrial processes as well.

Representation: For the extraction of invariant features, two autoencoder approaches, a stacked denoising autoencoder and a stacked convolutional autoencoder, were tested and evaluated. While both autoencoder converged to a reasonable reconstruction error, the stacked convolutional autoencoder performed better. A more important performance measure however, is the amount of information within the extracted features and their meaningfulness in terms of classification. To test this, the extracted features were fed into a support vector machine and used for classification. To establish a ground truth, the classification experiment was also conducted, using the raw images as input for a support vector machine. While the classification with the pure image turned out to perform best on the original data in terms of the F-score, this only holds for data without any variance. In a real process environment however, this will not be the case. In the following experiments variance, scaling from 0% up to 200% was introduced to evaluate the robustness against translational, rotational and scaling variance of both autoencoders. The stacked convolutional autoencoder performed significantly better than the stacked denoising autoencoder and the raw images. However, all approaches are not immune to variance. The classification performance in terms of the F-score decreased with increasing variance. When the variance crosses a certain threshold of significantly more variance than naturally occurs due to the setup process, the classification performance is no longer better than random guessing. It is interesting to note that rotational variance has a smaller influence than scaling or translation. The approach of using deep learning in order to extract robust features does work for a level of variance that naturally occurs during laser welding. The performance of the stacked autoencoders are as expected, the convolutional approach outperforming the fully connected one. More sophisticated architectures might prove even more robust to variance.

While training the autoencoder is a computational and time expensive procedure, it does not have to be computed locally. The autoencoder can be trained offline, using graphics card cluster and after learning convergence the weights can be copied to be used in a local data processing machine. This has several advantages: Expensive hardware does not have to be purchased, as computational time can be bought. In addition, using a central learning unit yields the possibility of collecting data from numerous different laser welding stations and therefore make use of all available data. Same goes for the learning of the predictions. Therefore only a few computations that are not expensive have to be performed locally. As a result, no expensive hardware has to be purchased to apply the suggested approach.

Knowledge and Prediction: To enable a closed loop control approach, a signal, containing the current performance, is essential. To provide this quality measure a general value temporal-difference approach, called nexting, was employed. The algorithm used the features, extracted by neural networks, and predicted the immediate quality as well as the quality within a short time horizon. The learned information is encoded in a 1000001 dimensional vector, only taking up a very small amount of memory but at the same time being transferable to different laser welding processes. This vector is originally initialized with all weights equal to 0. Therefore an unknown problem would be classified with a quality label of 0 as well, numerically indicating a non correct classification and even more important ensuring that no unknown low quality weld can accidentally be classified with a high quality. The evaluation results indicated that the features from a stacked convolutional autoencoder allowed a superior performance than the features from a stacked denoising autoencoder. Using the information within these features the algorithm was capable of not only correctly predicting the quality of the laser welding process, but also to detect errors during the process. An example was given for a grease contaminated welding seam. First, the learnability was demonstrated, learning and testing in the same weld. In a second step the algorithm was evaluated for making predictions about the immediate quality, using a leave one out cross validation. The experiments were completed by using the algorithm to predict the quality in the near future in a last experiment. This novel approach does not only allow closed loop control for laser welding for the first time, but is also transferable to different production processes, as general value functions can learn any learnable signal and are therefore not limited to specific applications. As the predictions can be made on different time scales and also into the future, corrective actions based on this signal could be learned and taken before the quality actually decreases. This would result not in a traditional closed loop control but in am preemptive control loop.

For both autoencoder approaches, the correct prediction of process 100 failed. The stacked denoising autoencoder additionally failed to correctly predict process 99. An inspection of the data set revealed that both processes were metal plates that have been contaminated with grease. A visual inspection of the provided features at the contaminated spots did not reveal a significant change of those features although the stacked convolutional autoencoding features clearly contain these information for process 99, as the nexting algorithm correctly predicts the change of quality for this process. The lack of generalization within the neural network provided features can most certainly come from a lack of sufficient training information for this problem. If more similar contaminated workpieces would be included in the training of the stacked autoencoders and the nexting algorithm as well, the problem would most certainly be identified more reliably.

As the weights for the nexting prediction algorithm were initialized at a value of zero, without learning the predicted quality would also correspond to zero. This initialization is vital due to the descending order of quality labels. It ensures that no weld will be higher predicted than its actual quality, ensuring no low quality weld will be used for critical structures. A weld that is represented by a completely unknown combination of features, provided by the autoencoders would therefore also be predicted with a zero quality label, being a clear indicator that human assistance is needed and additional training examples for this specific example are required.

Control: The introduced and evaluated actor-critic reinforcement learning algorithm was able to find the correct solution for the control problem within a short learning time. It furthermore demonstrated a robust and reliable performance as all trials converged to the same, correct solution. This is especially remarkable, as the algorithms choices were not fed directly into the system, but limited due to the systems capabilities, without the reinforcement learning agent knowing. The actions were mapped and clipped without feedback about this restriction to the algorithm. Furthermore, restrictions about the change in the laser power and the systems behaviour were imposed, but not known to the control algorithm. This particular result is a very strong indication that reinforcement learning is capable of finding the correct policy, without the need of providing exact prior knowledge about the process or the imposed restrictions. From a computational point of view the algorithm performed fast enough for real-time requirements, which corresponded to 1ms. Each control and learning iteration for the actor-critic reinforcement learning algorithm took 0.34 ms. The experiments so far were conducted using a simulated laser welding system, as no control access to a real laser was available. Despite the efforts of improving this simulation with the help of industrial laser welding experts, the algorithm could not demonstrate its potential in an integrated laser welding architecture. This is a reasonable choice for a first investigation into this problem. However, given correct information about the state, provided by a deep neural network and the correct information about the actual performance by the nexting approach, the actor-critic reinforcement learning algorithm should be able to perform as controller within the suggested architecture to establish closed loop control for laser welding processes and other processes for which the architecture can be utilized.

Both approaches (the adaptive PID controller and the intelligent architecture) demonstrate to outperform current control approaches or enable a closed loop control in the first place. The experiments resemble real live conditions and test not only the ability to solve the problem but also confront the control approaches with additional difficulties, e.g. unforeseen disturbances or extraordinary high variance in the setup. While the approaches are not implemented in real environments and machines, they demonstrate promising results for a variety of traditional and highly complex control problems.

6 Conclusion

This experiments in this thesis were conducted to answer important questions in production engineering. *How can data driven machine learning approaches be used for adaptive closed loop control and how well do they work?* And furthermore: *How can laser welding as an open loop control system be closed with a machine learning based feedback signal?* The experiments demonstrated possible working solutions for the research questions and evaluated their performance, either in simulation or on real-world data.

In summary, both approaches demonstrate the significant potential benefit that open and closed loop control system's can gain from applying machine learning. In the first part, a general dynamic neural network approach has been utilized to learn a data-driven way of adapting PID parameters for various control systems. Similar to a stochastic approach in adaptive control, the adaptive PID tuner does not rely on additional information about the system but only uses the same information as a regular PID controller does. This setting renders expensive hardware changes unnecessary. The general dynamic neural network does not only significantly improve the performance of the PID controller but also allows the control algorithm to perform better than model-based adaptive control algorithms both in terms of convergence and stability. This system has the capacity to deal with different kinds of control challenges that represent typical control challenges. It is furthermore able to deal with changing environmental conditions that might disturb the system. It therefore promises to address key requirements of modern control theory and modern industry. It is furthermore an important step towards autonomous and self-learning processes. It is the first combination of general dynamic neural networks and PID controller. As the setup is very intuitive and analogous to the PID controller and furthermore all tested neural networks demonstrated similar performance, no special expertise is necessary to include general dynamic neural networks in the way described within this work. The usage of general dynamic neural networks can therefore in fact be seen as a first step towards a unified stochastic approach for adaptive control for PID control, as the approach significantly outperforms the standard adaptive control algorithms in 13 out of 16 cases.

To the best of my knowledge, this work is the first usage of arbitrary general dynamic neural networks for PID parameter tuning. It is furthermore unique in its comprehensive evaluation and quantitative comparison, by using four completely different test systems that represent typical and challenging control tasks, including several different nonlinearities, instability, input delays or chaotic behavior.

The second part of this thesis contains a machine learning based approach to allow closed loop control for complex industrial systems, such as laser welding systems, that can so far only be controlled in an open loop. By combining different algorithms from the field of deep learning and reinforcement learning, an intelligent architecture is created that resembles human capabilities to cope with this complex challenge. Deep autoencoders are used to provide an invariant representation for the problem at hand in the form of low dimensional features. Those features are used in a general value function algorithm to generate quality predictions on different time scales. Such quality predictions can be used as a control signal and therefore do not only allow closed loop control for the first time but also yield a preemptive control due to their temporal extension. Together, the features and the predictions allow closed loop control via a policy gradient reinforcement learning algorithm that can learn to apply the correct power without human assistance.

The combination of those algorithms into one architecture is a novel approach that not only is the first to use deep learning for laser welding but also to combine deep learning with general value functions [72]. As all algorithms work in terms of goals instead of mechanism, the approach can easily be transferred and utilized to other open loop settings. The data-driven learning and adapting empowers it to work in a self-learning and self-optimizing way that can adapt to changing conditions. It therefore promises to address key requirements of modern industry, in a way that our architecture combines fast learning with the capability to work independently from human teaching. This makes the present thesis an important contribution to not only industrial control engineering and adaptive control, but also the study of intelligent systems and machine intelligence. It also demonstrates, how industrial control approaches can benefit from the extension by machine learning algorithms. Together, the introduced machine learning approaches yield great potential to further automatize production engineering and dramatically reduce production costs.

Bibliography

1. K. J. Åström, *Introduction to stochastic control theory*. Courier Corporation, 2012.

2. T. Wuest, D. Weimer, C. Irgens, and K.-D. Thoben, “Machine learning in manufacturing: advantages, challenges, and applications,” *Production & Manufacturing Research*, vol. 4, no. 1, pp. 23–45, 2016.

3. K. H. Ang, G. Chong, and Y. Li, “PID control system analysis, design, and technology,” *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559–576, 2005.

4. Y. Xia, Z. Zhu, M. Fu, and S. Wang, “Attitude tracking of rigid spacecraft with bounded disturbances,” *IEEE Transactions on Industrial Electronics*, vol. 58, no. 2, pp. 647–659, 2011.

5. M. Safonov and M. Athans, “Gain and phase margin for multiloop lqg regulators,” *IEEE Transactions on Automatic Control*, vol. 22, no. 2, pp. 173–179, 1977.

6. E. Davison, “The robust control of a servomechanism problem for linear time-invariant multivariable systems,” *IEEE Transactions on Automatic Control*, vol. 21, no. 1, pp. 25–34, 1976.

7. G. Zames, “Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses,” *IEEE Transactions on Automatic Control*, vol. 26, no. 2, pp. 301–320, 1981.

8. S. Xu, T. Chen, *et al.*, “Robust h-infinity control for uncertain stochastic systems with state delay,” *IEEE Transactions on Automatic Control*, vol. 47, no. 12, pp. 2089–2094, 2002.

9. F. Li, P. Shi, X. Wang, and H. R. Karimi, “Robust h-infinity filtering for networked control systems with markovian jumps and packet dropouts,” *Modeling, Identification and Control*, vol. 35, no. 3, p. 159, 2014.

10. W.-J. Cao and J.-X. Xu, “Nonlinear integral-type sliding surface for both matched and unmatched uncertain systems,” *IEEE Transactions on Automatic Control*, vol. 49, no. 8, pp. 1355–1360, 2004.

11. M. Rubagotti, D. M. Raimondo, A. Ferrara, and L. Magni, “Robust model predictive control with integral sliding mode in continuous-time sampled-data nonlinear systems,” *IEEE Transactions on Automatic Control*, vol. 56, no. 3, pp. 556–570, 2011.

12. J. Yang, S. Li, and X. Yu, “Sliding-mode control for systems with mismatched uncertainties via a disturbance observer,” *IEEE Transactions on Industrial Electronics*, vol. 60, no. 1, pp. 160–169, 2013.

13. E. Feron, P. Apkarian, and P. Gahinet, “Analysis and synthesis of robust control systems via parameter-dependent lyapunov functions,” *IEEE Transactions on Automatic Control*, vol. 41, no. 7, pp. 1041–1046, 1996.

14. D. M. Adhyaru and A. K. Shah, “Multiple lyapunov based robust control for the nonlinear hybrid dynamical systems,” in *Proceedings of the International Conference on Intelligent Systems and Control*, pp. 1–6, IEEE, 2016.

15. D. Shishika, J. K. Yim, and D. A. Paley, “Robust lyapunov control design for bioinspired pursuit with autonomous hovercraft,” *IEEE Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 509–520, 2017.

16. Y. Zhou, M. Chen, and C. Jiang, “Robust tracking control of uncertain mimo nonlinear systems with application to uavs,” *IEEE/CAA Journal of Automatica Sinica*, vol. 2, no. 1, pp. 25–32, 2015.

17. M. P. Gryning, Q. Wu, M. Blanke, H. H. Niemann, and K. P. Andersen, “Wind turbine inverter robust loop-shaping control subject to grid interaction effects,” *IEEE Transactions on Sustainable Energy*, vol. 7, no. 1, pp. 41–50, 2016.

18. J. Pulkkinen, H. N. Koivo, and K. Makela, “Tuning of a robust pid controller-application to heating process in extruder,” in *Proceedings of the International Conference on Control Applications*, pp. 811–816, IEEE, 1993.

19. S. Vachirasricirikul and I. Ngamroo, “Robust lfc in a smart grid with wind power penetration by coordinated v2g control and frequency controller,” *IEEE Transactions on Smart Grid*, vol. 5, no. 1, pp. 371–380, 2014.

20. C.-M. Lin, C.-M. Lin, C.-H. Chen, and D. Yeung, “Robust pid control system design for chaotic systems using particle swarm optimization algorithm,” in *Proceedings of the International Conference on Machine Learning and Cybernetics*, vol. 6, pp. 3294–3299, 2009.

21. J. Aseltine, A. Mancini, and C. Sarture, “A survey of adaptive control systems,” *IRE Transactions on Automatic Control*, vol. 6, no. 1, pp. 102–108, 1958.

22. S. Sastry and M. Bodson, *Adaptive control: stability, convergence and robustness.* Courier Corporation, 2011.

23. H. Kakigano, Y. Miura, and T. Ise, “Distribution voltage control for dc microgrids using fuzzy control and gain-scheduling technique,” *IEEE Transactions on Power Electronics*, vol. 28, no. 5, pp. 2246–2258, 2013.

24. H. Zhang, X. Zhang, and J. Wang, “Robust gain-scheduling energy-to-peak control of vehicle lateral dynamics stabilisation,” *Vehicle System Dynamics*, vol. 52, no. 3, pp. 309–340, 2014.

25. H. Zhang and J. Wang, “Vehicle lateral dynamics control through afs/dyc and robust gain-scheduling approach,” *IEEE Transactions on Vehicular Technology*, vol. 65, no. 1, pp. 489–494, 2016.

26. M. CORLESS, “Simple adaptive controllers for systems which are stabilizable via high gain feedback,” *IMA Journal of Mathematical Control and Information*, vol. 8, no. 4, pp. 379–387, 1991.

27. A. Ilchmann and S. Townley, “Simple adaptive stabilization of high-gain stabilizable systems,” *Systems & Vontrol Letters*, vol. 20, no. 3, pp. 189–198, 1993.

28. X. Ye, “Switching adaptive output-feedback control of nonlinearly parametrized systems,” *Automatica*, vol. 41, no. 6, pp. 983–989, 2005.

29. H. Schuster, C. Westermaier, and D. Schroder, “Non-identifier-based adaptive control for a mechatronic system achieving stability and steady state accuracy,” in *Proceedings of the International Conference on Computer Aided Control System Design, Control Applications & Intelligent Control*, pp. 1819–1824, IEEE, 2006.

30. P. Parks, “Liapunov redesign of model reference adaptive control systems,” *IEEE Transactions on Automatic Control*, vol. 11, no. 3, pp. 362–367, 1966.

31. D. DesForges, “The application of model-referenced adaptive control to robotic manipulators,” *Journal of Dynamic Systems, Measurement and Control*, vol. 101, p. 193, 1979.

32. X.-H. Yu and W. W. Recker, “Stochastic adaptive control model for traffic signal systems,” *Transportation Research Part C: Emerging Technologies*, vol. 14, no. 4, pp. 263–282, 2006.

33. M. Athans, D. Castanon, K.-P. Dunn, C. Greene, W. Lee, N. Sandell, and A. Willsky, “The stochastic control of the f-8c aircraft using a multiple model adaptive control (mmac) method–part i: Equilibrium flight,” *IEEE Transactions on Automatic Control*, vol. 22, no. 5, pp. 768–780, 1977.

34. R. E. Kalman, “Design of a self-optimising control system,” *ASME Transactions*, vol. 80, pp. 468–478, 1958.

35. H. Koivo, “A multivariable self-tuning controller,” *Automatica*, vol. 16, no. 4, pp. 351–366, 1980.

36. Y. A.-R. I. Mohamed and E. F. El-Saadany, “An improved deadbeat current control scheme with a novel adaptive self-tuning load model for a three-phase pwm voltage-source inverter,” *IEEE Transactions on Industrial Electronics*, vol. 54, no. 2, pp. 747–759, 2007.

37. S. Soyguder, M. Karakose, and H. Alli, “Design and simulation of self-tuning pid-type fuzzy adaptive control for an expert hvac system,” *Expert Systems with Applications*, vol. 36, no. 3, pp. 4566–4573, 2009.

38. O. Berger-Tal, J. Nathan, E. Meron, and D. Saltz, “The exploration-exploitation dilemma: a multidisciplinary framework,” *PLOS One*, vol. 9, no. 4, p. e95693, 2014.

39. J. Wieslander and B. Wittenmark, “An approach to adaptive control using real time identification,” *Automatica*, vol. 7, no. 2, pp. 211–217, 1971.

40. P. Kumar, “A survey of some results in stochastic adaptive control,” *SIAM Journal on Control and Optimization*, vol. 23, no. 3, pp. 329–380, 1985.

41. L. Guo, “Self-convergence of weighted least-squares with applications to stochastic adaptive control,” *IEEE Transactions on Automatic Control*, vol. 41, no. 1, pp. 79–89, 1996.

42. J. P. Hespanha, D. Liberzon, and A. S. Morse, “Overcoming the limitations of adaptive control by means of logic-based switching,” *Systems & control letters*, vol. 49, no. 1, pp. 49–65, 2003.

43. P. Zipkin, S. S. of Management, and M. I. of Technology, *The Limits of Mass Customization*. MIT Sloan Management Review, 2001.

44. F. Lin, S. Lin, and S. Zeng, “Co-simulation of neural networks PID control for ship steering hydraulic system,” in *Proceedings of the International Conference on Information and Automation*, pp. 2097–2100, 2010.

45. B. V. Murthy, Y. P. Kumar, and U. R. Kumari, “Application of neural networks in process control: automatic/online tuning of pid controller gains for±10% disturbance rejection,” in *Proceedings of the International Conference on Advanced Communication Control and Computing Technologies*, pp. 348–352, IEEE, 2012.

46. S.-G. Cui, H.-L. Pan, and J.-G. Li, “Application of self-tuning of PID control based on BP neural networks in the mobile robot target tracking,” in *Proceedings of the International Conference on Instrumentation, Measurement, Computer, Communication and Control*, pp. 1574–1577, 2013.

47. S. Omatu, T. Fujinaka, and M. Yoshioka, “Neuro-PID control for inverted single and double pendulums,” in *Proceedings of the International Conference on Systems, Man, and Cybernetics*, vol. 4, pp. 2685–2690, IEEE, 2000.

48. D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, “Why does unsupervised pre-training help deep learning?,” *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 625–660, 2010.

49. B. Guo, L. Hu, and Y. Bai, “PMSM servo system based on dynamic recurrent neural networks PID controller,” in *Proceedings of the International Conference on Power Electronics and Motion Control*, vol. 4, pp. 2417–2421, 2012.

50. Z.-Y. Song, C.-Y. Liu, and X.-L. Song, “Application of self-tuning pid control based on diagonal recurrent neural network in crystallization process,” in *Proceedings of the International Conference on Machine Learning and Cybernetics*, pp. 365–369, 2006.

51. C.-C. Tsai and Y.-L. Chang, “Self-tuning PID control using recurrent wavelet neural networks,” in *Proceedings of the International Conference on Systems, Man, and Cybernetics*, pp. 3111–3116, IEEE, Oct. 2012.

52. M.-g. Zhang, X.-G. Wang, and W.-h. Li, “The self-tuning PID decoupling control based on the diagonal recurrent neural network,” in *Proceedings of the International Conference on Machine Learning and Cybernetics*, pp. 3016–3020, 2006.

53. K. Zhang and X. An, “Design of multivariable self-tuning PID controllers via quasi-diagonal recurrent wavelet neural network,” in *Proceedings of the International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 2, pp. 95–99, 2010.

54. D. You, X. Gao, and S. Katayama, “Wpd-pca-based laser welding process monitoring and defects diagnosis by using fnn and svm,” *IEEE Transactions on Industrial Electronics*, vol. 62, no. 1, pp. 628–636, 2015.

55. T. Wang, J. Chen, X. Gao, and W. Li, “Quality monitoring for laser welding based on high-speed photography and support vector machine,” *Applied Sciences*, vol. 7, no. 3, p. 299, 2017.

56. Y. Zou, D. Du, and G. Guo, “Image matching for weld seam tracking based on non-linear dimensionality reduction method isomap,” in *Proceedings of the International Conference on Fuzzy Systems and Knowledge Discovery*, vol. 6, pp. 2826–2829, IEEE, 2010.

57. W. Chen, S. Pang, D. Liao, and J. Zhou, “Numerical simulation of transient keyhole instability and weld pool behaviors in parallel dual-beam laser welding part i. model development and transient keyhole behaviors,” *Transactions of the China Welding Institution*, vol. 3, p. 001, 2013.

58. D. U. Chang, “Laser weld quality monitoring system,” Oct. 28 1997. US Patent 5,681,490.

59. D. Westerbaan, D. Parkes, S. Nayak, D. Chen, E. Biro, F. Goodwin, and Y. Zhou, “Effects of concavity on tensile and fatigue properties in fibre laser welding of automotive steels,” *Science and Technology of Welding and Joining*, vol. 19, no. 1, pp. 60–68, 2014.

60. A. C. A. Asséko, B. Cosson, F. Schmidt, Y. Le Maoult, R. Gilblas, and E. Lafranche, “Laser transmission welding of composites–part B: Experimental validation of numerical model,” *Infrared Physics & Technology*, vol. 73, pp. 304–311, 2015.

61. F. Liu, C. Quek, and G.-S. Ng, “Neural network model for time series prediction by reinforcement learning,” in *Proceedings of the International Conference on Neural Networks*, vol. 2, pp. 809–814, 2005.

62. I. Stork gennant Wersborg, *A Cognitive Architecture For Production Systems such as Laser Material Processing*. Dissertation, Technische Universität München, München, 2010.

63. M. Jager, S. Humbert, and F. Hamprecht, “Sputter tracking for the automatic monitoring of industrial laser-welding processes,” *IEEE Transactions on Industrial Electronics*, vol. 55, no. 5, pp. 2177–2184, 2008.

64. C. Alippi, P. Braione, V. Piuri, and F. Scotti, “A methodological approach to multisensor classification for innovative laser material processing units,” in *Proceedings of the International Conference on Instrumentation and Measurement Technology*, vol. 3, pp. 1762–1767, 2001.

65. A. Ancona, V. Spagnolo, P. M. Lugarà, and M. Ferrara, “Optical sensor for real-time monitoring of CO_2 laser welding process,” *Applied Optics*, vol. 40, no. 33, pp. 6019–6025, 2001.

66. G. Schroth, I. Stork, G. Wersborg, and K. Diepold, “A cognitive system for autonomous robotic welding,” in *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 3148–3153, 2009.

67. J. Günther, P. Pilarski, G. Helfrich, H. Shen, and K. Diepold, “First steps towards an intelligent laser welding architecture using deep neural networks and reinforcement learning,” in *Proceedings of the International Conference on System-Integrated Intelligence: Challenges for Product and Production Engineering*, pp. 474–483, Jul 2014.

68. J. Shao and Y. Yan, “Review of techniques for on-line monitoring and inspection of laser welding,” *Journal of Physics: Conference Series*, vol. 15, no. 1, pp. 101–107, 2005.

69. J. Beersiek, “A CMOS camera as tool for process analysis not only for laser beam welding,” in *Proceedings of the International Conference on Applications of Lasers & Electro-Optics*, pp. 1185–1193, 2001.

70. C. Ho, J. Yu, and Y. Tsai, “Thermal characteristics of region surrounding laser welding keyhole,” in *Proceedings of the International Conference on Consumer Electronics*, pp. 1–2, IEEE, 2016.

71. J. Kroos, U. Gratzke, M. Vicanek, and G. Simon, “Dynamic behaviour of the keyhole in laser welding,” *Journal of Physics D: Applied Physics*, vol. 26, no. 3, pp. 481–486, 1993.

72. J. Günther, P. M. Pilarski, G. Helfrich, H. Shen, and K. Diepold, “Intelligent laser welding through representation, prediction, and control learning: An architecture with deep neural networks and reinforcement learning,” *Mechatronics*, vol. 34, pp. 1–11, 2016.

73. O. Föllinger, *Regelungstechnik: Einführung in d. Methoden u. ihre Anwendung*. Elitera, 1980.

74. L. Ljung, *System identification*. Wiley Online Library, 1999.

75. S. Engell and F. Allgöwer, *Entwurf nichtlinearer Regelungen*. Oldenbourg, 1995.

76. R. Hermann and A. Krener, “Nonlinear controllability and observability,” *IEEE Transactions on Automatic Control*, vol. 22, no. 5, pp. 728–740, 1977.

77. N. Minorsky, “Directional stability of automatically steered bodies,” *Naval Engineers Journal*, vol. 42, no. 2, pp. 280–309, 1922.

78. K. J. Åström and T. Hägglund, “Pid controllers: theory, design, and tuning,” *Instrument Society of America*, 1995.

79. R. Gorez, “A survey of pid auto-tuning methods: Feature issue controller tuning,” *Journal A*, vol. 38, no. 1, pp. 3–10, 1997.

80. J. Quevedo and T. Escobet, *Digital control: Past, present and future of PID control*. Elsevier Science Inc., 2000.

81. J. Ziegler and N. Nichols, “Optimum settings for automatic controllers,” *ASME Transactions*, vol. 64, no. 11, pp. 759–768, 1942.

82. G. Winter, J. Periaux, M. Galan, and P. Cuesta, *Genetic algorithms in engineering and computer science*. John Wiley & Sons, Inc., 1996.

83. E. A. Gargari, F. Hashemzadeh, R. Rajabioun, and C. Lucas, "Colonial competitive algorithm: A novel approach for PID controller design in MIMO distillation column process," *International Journal of Intelligent Computing and Cybernetics*, vol. 1, no. 3, pp. 337–355, 2008.

84. T. Slavov and O. Roeva, "Application of genetic algorithm to tuning a PID controller for glucose concentration control," *WSEAS Transactions on Systems and Control*, vol. 11, no. 7, pp. 223–233, 2012.

85. T. Back, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

86. E. Alpaydin, *Introduction to machine learning*. MIT press, 2014.

87. T. M. Mitchell, "Machine learning. 1997," *Burr Ridge, IL: McGraw Hill*, vol. 45, no. 37, pp. 870–877, 1997.

88. D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Computation*, vol. 8, no. 7, pp. 1341–1390, 1996.

89. S. Thrun, "Learning to play the game of chess," *Advances in Neural Information Processing Systems*, vol. 7, 1995.

90. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

91. D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Proceedings of the International Conference on Advances in Neural Information Processing Systems Conference*, pp. 2843–2851, 2012.

92. J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, vol. 32, pp. 323–332, 2012.

93. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

94. F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

95. W. Schultz, "Predictive reward signal of dopamine neurons," *Journal of Neurophysiology*, vol. 80, no. 1, pp. 1–27, 1998.

96. A. Khotanzad and J.-H. Lu, “Classification of invariant image representations using a neural network,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 6, pp. 1028–1038, 1990.

97. S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, “Face recognition: A convolutional neural-network approach,” *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.

98. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proceedings of the International Conference on Advances in Neural Information Processing Systems Conference*, pp. 1106–1114, 2012.

99. R. Caruana and A. Niculescu-Mizil, “An empirical comparison of supervised learning algorithms,” in *Proceedings of the International Conference on Machine Learning*, pp. 161–168, 2006.

100. S. Tong and D. Koller, “Support vector machine active learning with applications to text classification,” *Journal of machine learning research*, vol. 2, no. Nov, pp. 45–66, 2001.

101. M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “Xnor-net: Imagenet classification using binary convolutional neural networks,” in *Proceedings of the European Conference on Computer Vision*, pp. 525–542, 2016.

102. B. K. Samanthula, Y. Elmehdwi, and W. Jiang, “K-nearest neighbor classification over semantically secure encrypted relational data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1261–1273, 2015.

103. M. A. Friedl and C. E. Brodley, “Decision tree classification of land cover from remotely sensed data,” *Remote Sensing of Environment*, vol. 61, no. 3, pp. 399–409, 1997.

104. K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

105. T. Hastie, R. Tibshirani, and J. Friedman, *Overview of supervised learning*. Springer, 2009.

106. I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

107. S. Hawe, M. Seibert, and M. Kleinsteuber, “Separable dictionary learning,” in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 438–445, 2013.

108. P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local

denoising criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

109. A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 215–223, 2011.

110. S. Ghosh and S. K. Dubey, "Comparative analysis of k-means and fuzzy c-means algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 4, pp. 35–39, 2013.

111. S. Lee and M. M. Crawford, "Unsupervised multistage image classification using hierarchical clustering with a bayesian similarity measure," *IEEE Transactions on Image Processing*, vol. 14, no. 3, pp. 312–320, 2005.

112. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

113. A. Cotter, O. Shamir, N. Srebro, and K. Sridharan, "Better mini-batch algorithms via accelerated gradient methods," in *Proceedings of the International Conference on Advances in Neural Information Processing Systems*, pp. 1647–1655, 2011.

114. Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade*, pp. 437–478, Springer, 2012.

115. I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the International Conference on Machine Learning*, vol. 28, pp. 1139–1147, JMLR Workshop and Conference Proceedings, 2013.

116. A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Proceedings of the International Conference on Advances in Neural Information Processing Systems*, vol. 4, pp. 950–957, 1991.

117. J. Moody, S. Hanson, A. Krogh, and J. A. Hertz, "A simple weight decay can improve generalization," *Advances in Neural Information Processing Systems*, vol. 4, pp. 950–957, 1995.

118. X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, vol. 15, pp. 315–323, 2011.

119. M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, *et al.*, "On rectified linear units for speech processing," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 3517–3521, IEEE, 2013.

120. B. A. Olshausen, "Highly overcomplete sparse coding," in *Proceedings of the International Conference on Imaging, Sensors and Applications*, pp. 86510S–86510S, International Society for Optics and Photonics, 2013.

121. J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional autoencoders for hierarchical feature extraction," *Artificial Neural Networks and Machine Learning*, pp. 52–59, 2011.

122. B. Du, W. Xiong, J. Wu, L. Zhang, L. Zhang, and D. Tao, "Stacked convolutional denoising auto-encoders for feature representation," *IEEE Transactions on Cybernetics*, vol. 47, no. 4, pp. 1017–1027, 2017.

123. D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of Physiology*, vol. 195, no. 1, pp. 215–243, 1968.

124. M. S. Nixon and A. S. Aguado, *Feature extraction & image processing for computer vision*. Academic Press, 2012.

125. T. Serre, G. Kreiman, M. Kouh, C. Cadieu, U. Knoblich, and T. Poggio, "A quantitative theory of immediate visual recognition," *Progress in Brain Research*, vol. 165, pp. 33–56, 2007.

126. J. Schmidhuber, M. Eldracher, and B. Foltin, "Semilinear predictability minimization produces well-known feature detectors," *Neural Computation*, vol. 8, no. 4, pp. 773–786, 1996.

127. P. O. Hoyer and A. Hyvärinen, "Independent component analysis applied to feature extraction from colour and stereo images," *Network: Computation in Neural Systems*, vol. 11, no. 3, pp. 191–210, 2000.

128. D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *Proceedings of the International Conference on Artificial Intelligence*, 2011.

129. O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: Lessons learned from the 2015 mscoco image captioning challenge," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 652–663, 2017.

130. B. Leng, S. Guo, X. Zhang, and Z. Xiong, "3d object retrieval with stacked local convolutional autoencoder," *Signal Processing*, vol. 112, pp. 119–128, 2015.

131. R. Kothari and K. Agyepong, "On lateral connections in feed-forward neural networks," in *Proceedings of the International Conference on Neural Networks*, vol. 1, pp. 13–18, IEEE, 1996.

132. M. I. Jordan, “Chapter 25 - serial order: A parallel distributed processing approach,” in *Advances in Psychology* (J. W. D. a. V. P. Dorsel, ed.), vol. 121 of *Neural-Network Models of Cognition Biobehavioral Foundations*, pp. 471–495, North-Holland, 1997.

133. N. Karunanithi, D. Whitley, and Y. K. Malaiya, “Using neural networks in reliability prediction,” *IEEE Software*, vol. 9, no. 4, pp. 53–59, 1992.

134. R. Yasdi, “Prediction of road traffic using a neural network approach,” *Neural Computing & Applications*, vol. 8, no. 2, pp. 135–142, 1999.

135. A. Baykasoğlu, T. Dereli, and S. Tanış, “Prediction of cement strength using soft computing techniques,” *Cement and Concrete Research*, vol. 34, no. 11, pp. 2083–2090, 2004.

136. J. L. Elman, “Finding structure in time,” *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

137. W. Junsong, W. Jiukun, Z. Maohua, and W. Junjie, “Prediction of internet traffic based on elman neural network,” in *Proceedings of the International Conference on Control and Decision Conference*, pp. 1248–1252, IEEE, 2009.

138. Z. Yang, M. Sun, W. Li, and W. Liang, “Modified elman network for thermal deformation compensation modeling in machine tools,” *The International Journal of Advanced Manufacturing Technology*, vol. 54, no. 5, pp. 669–676, 2011.

139. P. J. Werbos, “Backpropagation through time: what it does and how to do it,” *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

140. Y. Xia and J. Wang, “A recurrent neural network for solving nonlinear convex programs subject to linear constraints,” *IEEE Transactions on Neural Networks*, vol. 16, no. 2, pp. 379–386, 2005.

141. A. Graves and J. Schmidhuber, “Offline handwriting recognition with multidimensional recurrent neural networks,” in *Proceedings of the International Conference on Advances in Neural Information Processing Systems*, pp. 545–552, 2009.

142. R. S. Sutton, “Generalization in reinforcement learning: Successful examples using sparse coarse coding,” *Advances in Neural Information Processing Systems*, pp. 1038–1044, 1996.

143. P. Pilarski, T. Dick, and R. Sutton, “Real-time prediction learning for the simultaneous actuation of multiple prosthetic joints,” in *Proceedings of the International Conference on Rehabilitation Robotics*, pp. 1–8, 2013.

144. S. Schaal and C. G. Atkeson, “Robot juggling: implementation of memory-based learning,” *IEEE Control Systems*, vol. 14, no. 1, pp. 57–71, 1994.

145. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

146. R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1st ed., 1998.

147. A. White, *Developing a predictive approach to knowledge*. PhD thesis, University of Alberta, Edmonton, 2015.

148. C. M. Bishop, "Pattern recognition," *Machine Learning*, vol. 128, pp. 1–58, 2006.

149. L. Behlau, *Technology guide: Principles-applications-trends*. Springer Science & Business Media, 2009.

150. R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, *et al.*, "Policy gradient methods for reinforcement learning with function approximation.," in *Proceedings of the International Conference on Advances in Neural Information Processing Systems*, vol. 99, pp. 1057–1063, 1999.

151. R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora, "Fast gradient-descent methods for temporal-difference learning with linear function approximation," in *Proceedings of the International Conference on Machine Learning*, pp. 993–1000, ACM, 2009.

152. O. De Jesus and M. T. Hagan, "Backpropagation algorithms for a broad class of dynamic networks," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 14–27, 2007.

153. O. Nelles, *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. Springer Science & Business Media, 2013.

154. R. Isermann, *Identification of Dynamical Systems I (in German)*. Berlin, Germany: Springer-Verlag, 1988.

155. H. Pan, H. Wong, V. Kapila, and M. S. de Queiroz, "Experimental validation of a nonlinear backstepping liquid level controller for a state coupled two tank system," *Control Engineering Practice*, vol. 13, no. 1, pp. 27–40, 2005.

156. J.-J. Wang, "Simulation studies of inverted pendulum based on pid controllers," *Simulation Modelling Practice and Theory*, vol. 19, no. 1, pp. 440–449, 2011.

157. S. Tavakoli and M. Tavakoli, "Optimal tuning of PID controllers for first order plus time delay models using dimensional analysis," in *Proceedings of the International Conference on Control and Automation*, pp. 942–946, 2003.

158. D. M. Boskovic and M. Krstic, “Global stabilization of a thermal convection loop,” in *Proceedings of the International Conference on Decision and Control*, vol. 5, pp. 4777–4782, IEEE, 2000.

159. R. S. Smith and J. Doyle, “The two tank experiment: A benchmark control problem,” in *Proceedings of the American Control Conference*, pp. 2026–2031, IEEE, 1988.

160. M. Majstorovic, I. Nikolic, J. Radovic, and G. Kvascev, “Neural network control approach for a two-tank system,” in *Proceedings of the International Conference on Neural Network Applications in Electrical Engineering*, pp. 215–218, IEEE, 2008.

161. T. Takagi, I. Mizumoto, and J. Tsunematsu, “Performance-driven adaptive output feedback control and its application to two-tank system,” in *Proceedings of the International Conference on Advanced Mechatronic Systems*, pp. 254–259, IEEE, 2014.

162. R. Lozano and B. Brogliato, “Adaptive control of robot manipulators with flexible joints,” *IEEE Transactions on Automatic Control*, vol. 37, no. 2, pp. 174–181, 1992.

163. W. Li, H. Ding, and K. Cheng, “An investigation on the design and performance assessment of double-PID and LQR controllers for the inverted pendulum,” in *Proceedings of the International Conference on Control*, pp. 190–196, 2012.

164. K. J. Åström and B. Wittenmark, *Adaptive control*. Courier Corporation, 2013.

165. J. A. Jaleel and R. M. Francis, “Simulated annealing based control of an inverted pendulum system,” in *Proceedings of the International Conference on Control Communication and Computing*, pp. 204–209, IEEE, 2013.

166. C. Wang, W. Fu, and Y. Shi, “Fractional order proportional integral differential controller design for first order plus time delay system,” in *Proceedings of the International Conference on Control and Decision Conference*, pp. 3259–3262, IEEE, May 2013.

167. F. Merrikh-Bayat, “Optimal tuning rules of the fractional-order PID controllers with application to first-order plus time delay processes,” in *Proceedings of the International Symposium on Advanced Control of Industrial Processes*, pp. 403–408, 2011.

168. M. Stojic, M. Matijevic, and L. Draganovic, “A robust smith predictor modified by internal models for integrating process with dead time,” *IEEE Transactions on Automatic Control*, vol. 46, no. 8, pp. 1293–1298, 2001.

169. K. Warwick and D. Rees, *Industrial digital control systems*, vol. 37. IET, 1988.

170. X.-H. Long, B. Balachandran, and B. Mann, “Dynamics of milling processes with variable time delays,” *Nonlinear Dynamics*, vol. 47, no. 1, pp. 49–63, 2007.

171. A. L. Fradkov and R. J. Evans, “Control of chaos: Methods and applications in engineering,” *Annual Reviews in Control*, vol. 29, no. 1, pp. 33–56, 2005.

172. Y. Wang, J. Singer, and H. H. Bau, “Controlling chaos in a thermal convection loop,” *Journal of Fluid Mechanics*, vol. 237, pp. 479–498, 1992.

173. R. Vasquez and M. Krstic, “Thermal convection loop control by continuous backstepping and singular perturbations,” in *Proceedings of the American Control Conference*, pp. 3882–3887, IEEE, 2005.

174. R. Vazquez and M. Krstic, “Explicit integral operator feedback for local stabilization of nonlinear thermal convection loop pdes,” *Systems & Control Letters*, vol. 55, no. 8, pp. 624–632, 2006.

175. R. Vazquez and M. Krstic, “Boundary observer for output-feedback stabilization of thermal-fluid convection loop,” *IEEE Transactions on Control Systems Technology*, vol. 18, no. 4, pp. 789–797, 2010.

176. O. J. M. Smith, “Closed control of loops with dead time,” *Chemical Engineering Progress*, vol. 53, no. 5, pp. 217–219, 1957.

177. E. Reichensdörfer, J. Günther, and K. Diepold, “Recurrent neural networks for pid auto-tuning,” tech. rep., Technische Universität München, Lehrstuhl für Datenverarbeitung, 2017.

178. J. Günther, H. Shen, and K. Diepold, “Neural networks for fast sensor data processing in laser welding,” in *Jahreskolloquium - Bildverarbeitung in der Automation*, (Lemgo, Germany), Nov 2014.

179. Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

180. Y. Bengio, “Learning deep architectures for AI,” *Foundation and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

181. M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *Proceedings of the International Conference on Computer Vision*, pp. 818–833, Springer, 2014.

182. Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, *et al.*, “Greedy layer-wise training of deep networks,” *Advances in Neural Information Processing Systems*, vol. 19, p. 153, 2007.

183. G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

184. S. Lange, M. Riedmiller, and A. Voigtlander, "Autonomous reinforcement learning on raw visual input data in a real world application," in *Proceedings of the International Conference on Neural Networks*, pp. 1–8, 2012.

185. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," in *NIPS Deep Learning Workshop 2013*, 2013.

186. J. Gehring, Y. Miao, F. Metze, and A. Waibel, "Extracting deep bottleneck features using stacked auto-encoders," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 3377–3381, IEEE, 2013.

187. P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the International Conference on Machine Learning*, pp. 1096–1103, ACM, 2008.

188. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

189. D. Gilbert, *Stumbling on Happiness*. New York: Vintage, 2007.

190. A. Clark, "Whatever next? Predictive brains, situated agents, and the future of cognitive science," *Behavioral and Brain Sciences*, vol. 36, pp. 181–204, 2013.

191. R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup, "Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction," in *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, vol. 2, pp. 761–768, 2011.

192. J. Modayil, A. White, and R. S. Sutton, "Multi-timescale nexting in a reinforcement learning robot," *Adaptive Behavior*, vol. 22, no. 2, pp. 146–160, 2014.

193. J. Modayil, A. White, P. Pilarski, and R. Sutton, "Acquiring a broad range of empirical knowledge in real time by temporal-difference learning," in *Proceedings of the International Conference on Systems, Man, and Cybernetics*, pp. 1903–1910, 2012.

194. T. Degris, P. Pilarski, and R. Sutton, "Model-free reinforcement learning with continuous action in practice," in *Proceedings of the American Control Conference*, pp. 2177–2182, 2012.

195. P. Pilarski, M. Dawson, T. Degris, F. Fahimi, J. Carey, and R. Sutton, "Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning," in *Proceedings of the International Conference on Rehabilitation Robotics*, pp. 1–7, 2011.

196. H. Kimura, T. Yamashita, and S. Kobayashi, “Reinforcement learning of walking behavior for a four-legged robot,” in *Proceedings of the International Conference on Decision and Control*, vol. 1, pp. 411–416, 2001.

197. P. Thomas, M. Branicky, A. van den Bogert, and K. Jagodnik, “Application of the actor-critic architecture to functional electrical stimulation control of a human arm,” in *Proceedings of the International Conference on Innovative Applications of Artificial Intelligence Conference*, vol. 2009, pp. 165–172, 2009.

198. J. Peters and S. Schaal, “Natural actor-critic,” *Neurocomputing*, vol. 71, no. 7, pp. 1180–1190, 2008.

199. H. Min-ying, Shi-Junwei, C. jinjin Li-xin, and X. Yanqiu, “The research of welding parameters on weld shape in the laser deep penetration welding,” in *Proceedings of the International Conference on Mechanic Automation and Control Engineering*, pp. 3734–3737, 2010.

200. H. Shen, “Designing and training feedforward neural networks: A smooth optimisation perspective,” *arXiv preprint arXiv:1611.05827*, 2016.

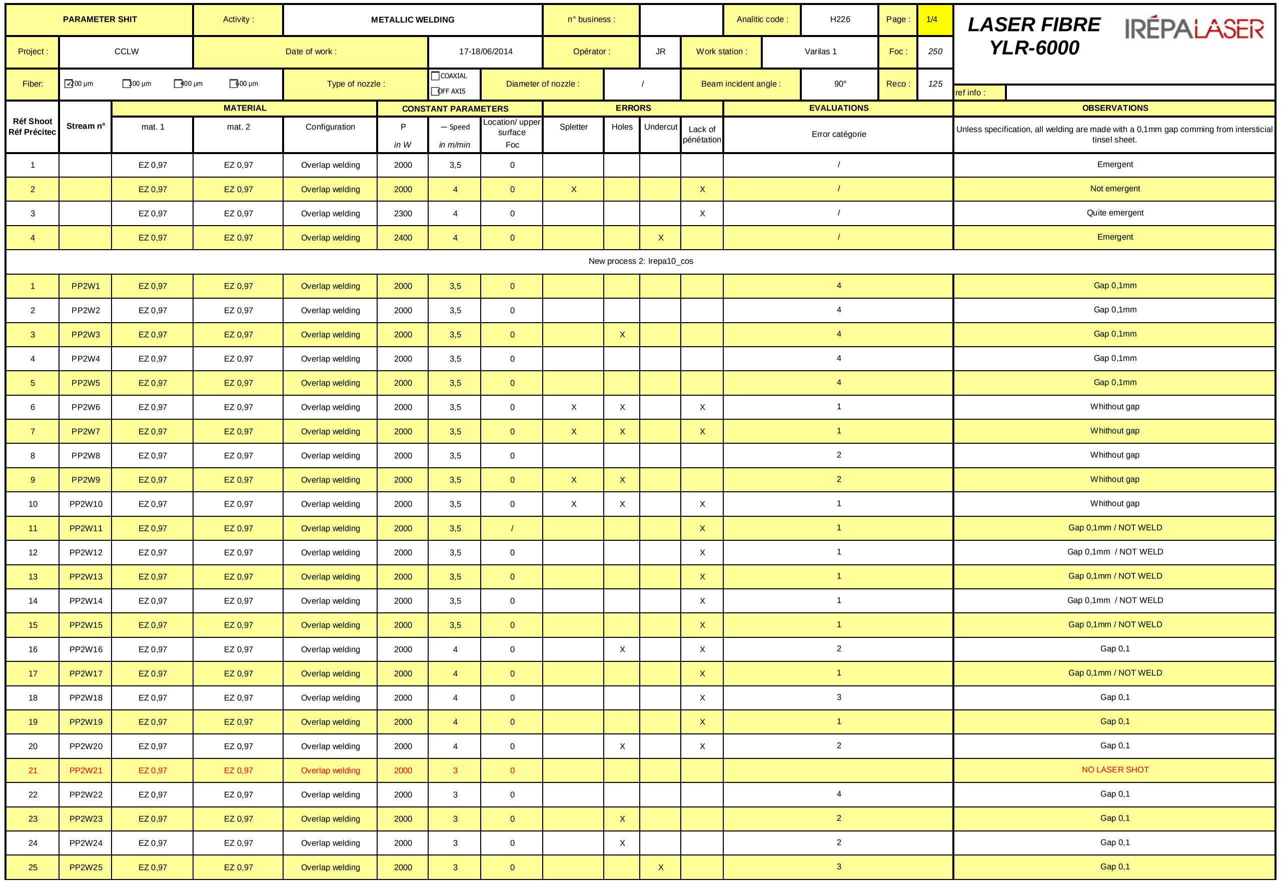
201. P. J. Angeline, G. M. Saunders, and J. B. Pollack, “An evolutionary algorithm that constructs recurrent neural networks,” *IEEE transactions on Neural Networks*, vol. 5, no. 1, pp. 54–65, 1994.

202. C. Sandner, J. Günther, and K. Diepold, “Automated optimization of dynamic neural network structure using genetic algorithms,” tech. rep., Technische Universität München, Lehrstuhl für Datenverarbeitung, 2017.

Appendix

A - Laser welding data sheets

As the machine learning algorithms in this thesis are mainly dependent on the provided data, this section provides the data sheets for the laser welding process P77. All workpieces were manually inspected and for each one a cross section was performed to investigate its quality and assign it to a quality class after EN ISO 13919-1:1996. The results of these investigations can be inspected in the following data sheets.



PARAMETER SHIT	Activity :	METALLIC WELDING	n° business :		Analitic code :	H226	Page :	1/4	
Project :	CCLW	Date of work :	17-18/06/2014	Opérator :	JR	Work station :	Varilas 1	Foc :	250
Fiber:	☑200 µm ☐300 µm ☐400 µm ☐600 µm	Type of nozzle :	☐COAXIAL ☐OFF AXIS	Diameter of nozzle :	/	Beam incident angle :	90°	Reco :	125

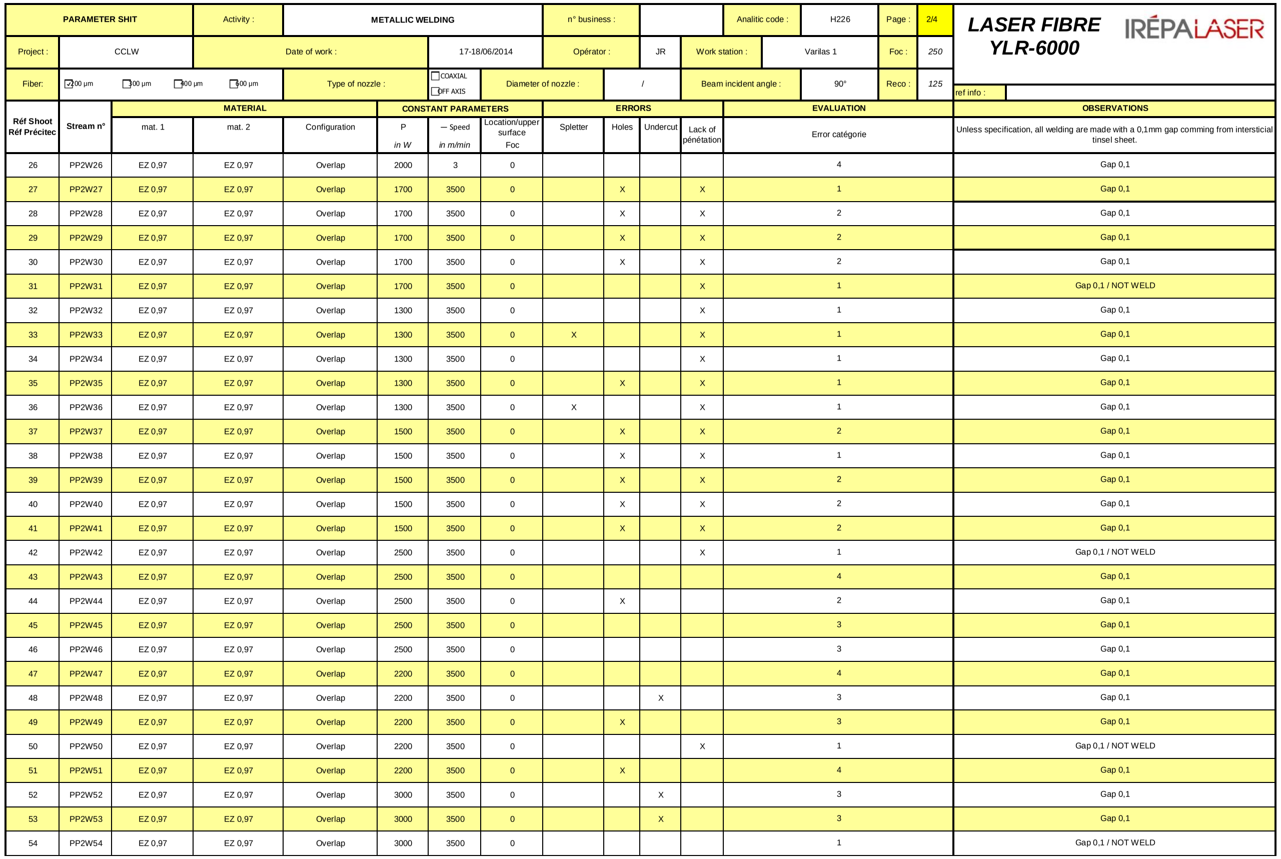
LASER FIBRE
YLR-6000

IRÉPALASER

ref info :

		MATERIAL			CONSTANT PARAMETERS			ERRORS				EVALUATIONS	OBSERVATIONS
Réf Shoot Réf Précitec	Stream n°	mat. 1	mat. 2	Configuration	P in W	— Speed in m/min	Location/ upper surface Foc	Spletter	Holes	Undercut	Lack of pénétation	Error catégorie	Unless specification, all welding are made with a 0,1mm gap comming from intersticial tinsel sheet.
1		EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0					/	Emergent
2		EZ 0,97	EZ 0,97	Overlap welding	2000	4	0	X			X	/	Not emergent
3		EZ 0,97	EZ 0,97	Overlap welding	2300	4	0				X	/	Quite emergent
4		EZ 0,97	EZ 0,97	Overlap welding	2400	4	0			X		/	Emergent
New process 2: Irepa10_cos													
1	PP2W1	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0					4	Gap 0,1mm
2	PP2W2	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0					4	Gap 0,1mm
3	PP2W3	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0		X			4	Gap 0,1mm
4	PP2W4	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0					4	Gap 0,1mm
5	PP2W5	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0					4	Gap 0,1mm
6	PP2W6	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0	X	X		X	1	Whithout gap
7	PP2W7	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0	X	X		X	1	Whithout gap
8	PP2W8	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0					2	Whithout gap
9	PP2W9	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0	X	X			2	Whithout gap
10	PP2W10	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0	X	X		X	1	Whithout gap
11	PP2W11	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	/				X	1	Gap 0,1mm / NOT WELD
12	PP2W12	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0				X	1	Gap 0,1mm / NOT WELD
13	PP2W13	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0				X	1	Gap 0,1mm / NOT WELD
14	PP2W14	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0				X	1	Gap 0,1mm / NOT WELD
15	PP2W15	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0				X	1	Gap 0,1mm / NOT WELD
16	PP2W16	EZ 0,97	EZ 0,97	Overlap welding	2000	4	0		X		X	2	Gap 0,1
17	PP2W17	EZ 0,97	EZ 0,97	Overlap welding	2000	4	0				X	1	Gap 0,1mm / NOT WELD
18	PP2W18	EZ 0,97	EZ 0,97	Overlap welding	2000	4	0				X	3	Gap 0,1
19	PP2W19	EZ 0,97	EZ 0,97	Overlap welding	2000	4	0				X	1	Gap 0,1
20	PP2W20	EZ 0,97	EZ 0,97	Overlap welding	2000	4	0		X		X	2	Gap 0,1
21	PP2W21	EZ 0,97	EZ 0,97	Overlap welding	2000	3	0						NO LASER SHOT
22	PP2W22	EZ 0,97	EZ 0,97	Overlap welding	2000	3	0					4	Gap 0,1
23	PP2W23	EZ 0,97	EZ 0,97	Overlap welding	2000	3	0		X			2	Gap 0,1
24	PP2W24	EZ 0,97	EZ 0,97	Overlap welding	2000	3	0		X			2	Gap 0,1
25	PP2W25	EZ 0,97	EZ 0,97	Overlap welding	2000	3	0			X		3	Gap 0,1

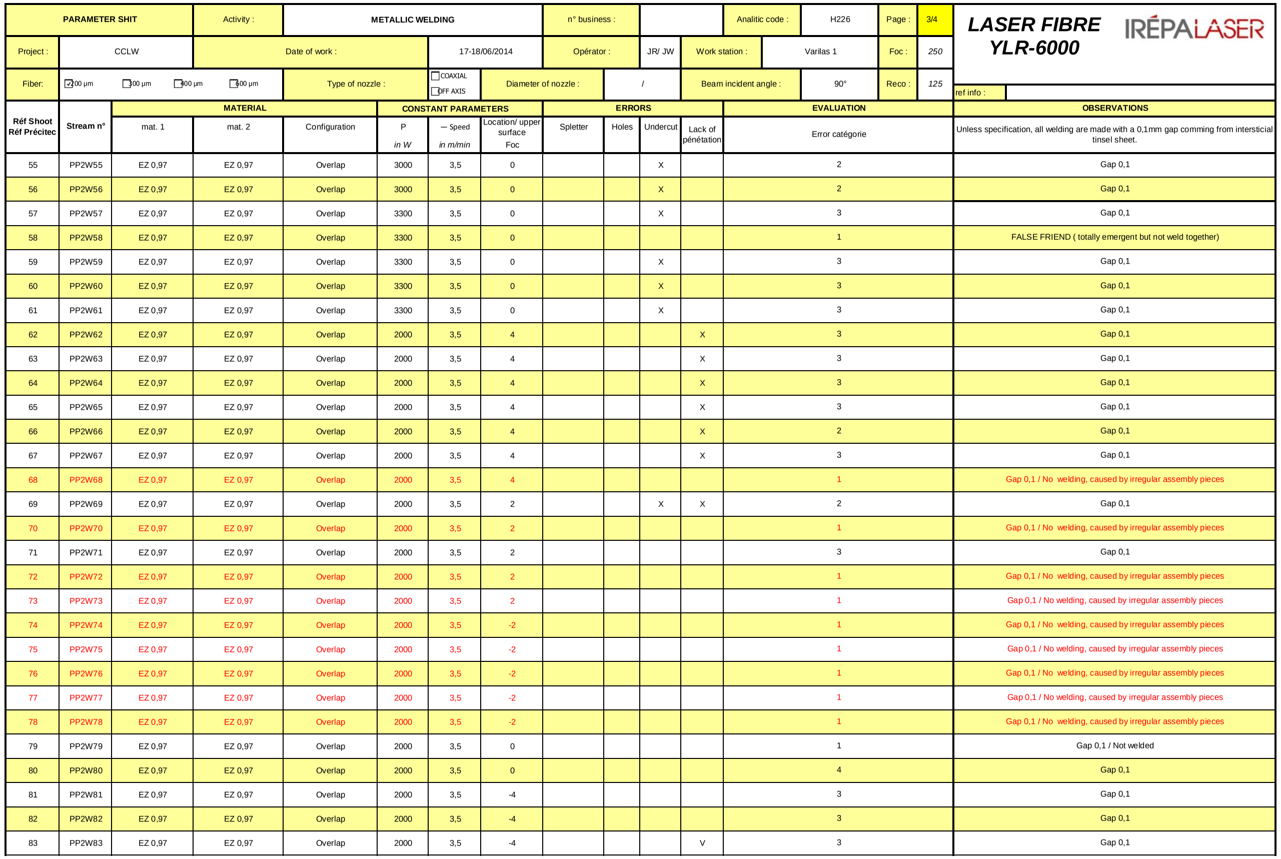
Figure 6.1: P77 Datasheet I.



PARAMETER SHIT	Activity :	METALLIC WELDING		n° business :		Analitic code :	H226	Page :	2/4	LASER FIBRE YLR-6000	IRÉPALASER
Project :	CCLW	Date of work :	17-18/06/2014	Opérator :	JR	Work station :	Varilas 1	Foc :	250		
Fiber:	☑200 µm □300 µm □400 µm □600 µm	Type of nozzle :	□COAXIAL □OFF AXIS	Diameter of nozzle :	/	Beam incident angle :	90°	Reco :	125	ref info :	

		MATERIAL			CONSTANT PARAMETERS			ERRORS				EVALUATION	OBSERVATIONS
Réf Shoot Réf Précitec	Stream n°	mat. 1	mat. 2	Configuration	P *in W*	— Speed *in m/min*	Location/upper surface Foc	Spletter	Holes	Undercut	Lack of pénétation	Error catégorie	Unless specification, all welding are made with a 0,1mm gap comming from intersticial tinsel sheet.
26	PP2W26	EZ 0,97	EZ 0,97	Overlap	2000	3	0					4	Gap 0,1
27	PP2W27	EZ 0,97	EZ 0,97	Overlap	1700	3500	0		X		X	1	Gap 0,1
28	PP2W28	EZ 0,97	EZ 0,97	Overlap	1700	3500	0		X		X	2	Gap 0,1
29	PP2W29	EZ 0,97	EZ 0,97	Overlap	1700	3500	0		X		X	2	Gap 0,1
30	PP2W30	EZ 0,97	EZ 0,97	Overlap	1700	3500	0		X		X	2	Gap 0,1
31	PP2W31	EZ 0,97	EZ 0,97	Overlap	1700	3500	0				X	1	Gap 0,1 / NOT WELD
32	PP2W32	EZ 0,97	EZ 0,97	Overlap	1300	3500	0				X	1	Gap 0,1
33	PP2W33	EZ 0,97	EZ 0,97	Overlap	1300	3500	0	X			X	1	Gap 0,1
34	PP2W34	EZ 0,97	EZ 0,97	Overlap	1300	3500	0				X	1	Gap 0,1
35	PP2W35	EZ 0,97	EZ 0,97	Overlap	1300	3500	0		X		X	1	Gap 0,1
36	PP2W36	EZ 0,97	EZ 0,97	Overlap	1300	3500	0	X			X	1	Gap 0,1
37	PP2W37	EZ 0,97	EZ 0,97	Overlap	1500	3500	0		X		X	2	Gap 0,1
38	PP2W38	EZ 0,97	EZ 0,97	Overlap	1500	3500	0		X		X	1	Gap 0,1
39	PP2W39	EZ 0,97	EZ 0,97	Overlap	1500	3500	0		X		X	2	Gap 0,1
40	PP2W40	EZ 0,97	EZ 0,97	Overlap	1500	3500	0		X		X	2	Gap 0,1
41	PP2W41	EZ 0,97	EZ 0,97	Overlap	1500	3500	0		X		X	2	Gap 0,1
42	PP2W42	EZ 0,97	EZ 0,97	Overlap	2500	3500	0				X	1	Gap 0,1 / NOT WELD
43	PP2W43	EZ 0,97	EZ 0,97	Overlap	2500	3500	0					4	Gap 0,1
44	PP2W44	EZ 0,97	EZ 0,97	Overlap	2500	3500	0		X			2	Gap 0,1
45	PP2W45	EZ 0,97	EZ 0,97	Overlap	2500	3500	0					3	Gap 0,1
46	PP2W46	EZ 0,97	EZ 0,97	Overlap	2500	3500	0					3	Gap 0,1
47	PP2W47	EZ 0,97	EZ 0,97	Overlap	2200	3500	0					4	Gap 0,1
48	PP2W48	EZ 0,97	EZ 0,97	Overlap	2200	3500	0			X		3	Gap 0,1
49	PP2W49	EZ 0,97	EZ 0,97	Overlap	2200	3500	0		X			3	Gap 0,1
50	PP2W50	EZ 0,97	EZ 0,97	Overlap	2200	3500	0				X	1	Gap 0,1 / NOT WELD
51	PP2W51	EZ 0,97	EZ 0,97	Overlap	2200	3500	0		X			4	Gap 0,1
52	PP2W52	EZ 0,97	EZ 0,97	Overlap	3000	3500	0			X		3	Gap 0,1
53	PP2W53	EZ 0,97	EZ 0,97	Overlap	3000	3500	0			X		3	Gap 0,1
54	PP2W54	EZ 0,97	EZ 0,97	Overlap	3000	3500	0					1	Gap 0,1 / NOT WELD

Figure 6.2: P77 Datasheet II.



LASER FIBRE
YLR-6000

IRÉPALASER

PARAMETER SHIT		Activity :	METALLIC WELDING		n° business :		Analitic code :	H226	Page :	3/4
Project :	CCLW	Date of work :	17-18/06/2014		Opérator :	JR/ JW	Work station :	Varilas 1	Foc :	250
Fiber:	☑200 µm ☐300 µm ☐400 µm ☐600 µm	Type of nozzle :	☐COAXIAL ☐OFF AXIS	Diameter of nozzle :	/		Beam incident angle :	90°	Reco :	125

ref info :

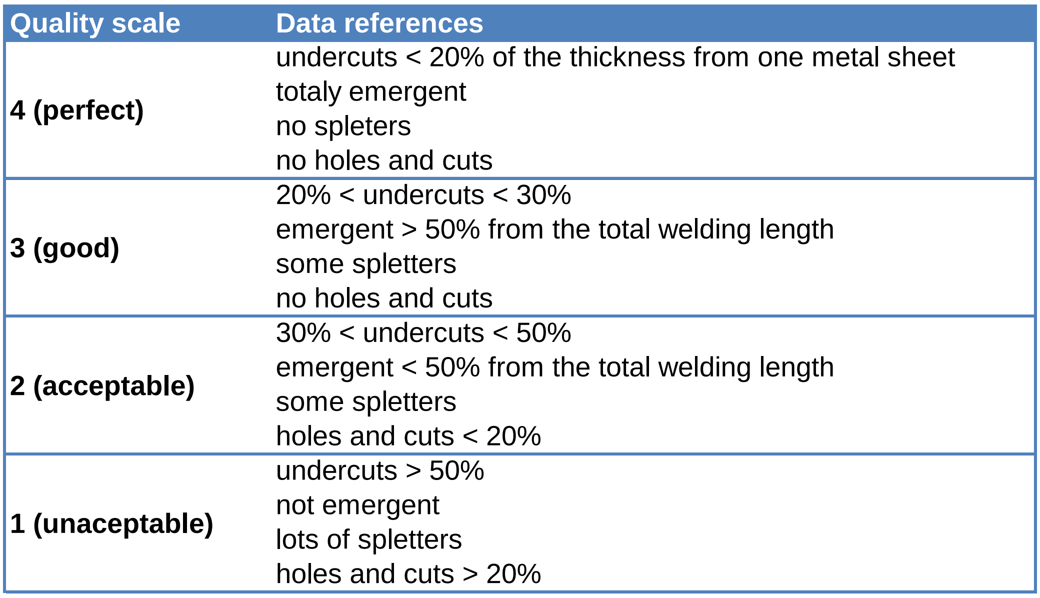
		MATERIAL			CONSTANT PARAMETERS			ERRORS				EVALUATION	OBSERVATIONS
Réf Shoot Réf Précitec	Stream n°	mat. 1	mat. 2	Configuration	P in W	— Speed in m/min	Location/ upper surface Foc	Spletter	Holes	Undercut	Lack of pénétation	Error catégorie	Unless specification, all welding are made with a 0,1mm gap comming from intersticial tinsel sheet.
55	PP2W55	EZ 0,97	EZ 0,97	Overlap	3000	3,5	0			X		2	Gap 0,1
56	PP2W56	EZ 0,97	EZ 0,97	Overlap	3000	3,5	0			X		2	Gap 0,1
57	PP2W57	EZ 0,97	EZ 0,97	Overlap	3300	3,5	0			X		3	Gap 0,1
58	PP2W58	EZ 0,97	EZ 0,97	Overlap	3300	3,5	0					1	FALSE FRIEND (totally emergent but not weld together)
59	PP2W59	EZ 0,97	EZ 0,97	Overlap	3300	3,5	0			X		3	Gap 0,1
60	PP2W60	EZ 0,97	EZ 0,97	Overlap	3300	3,5	0			X		3	Gap 0,1
61	PP2W61	EZ 0,97	EZ 0,97	Overlap	3300	3,5	0			X		3	Gap 0,1
62	PP2W62	EZ 0,97	EZ 0,97	Overlap	2000	3,5	4				X	3	Gap 0,1
63	PP2W63	EZ 0,97	EZ 0,97	Overlap	2000	3,5	4				X	3	Gap 0,1
64	PP2W64	EZ 0,97	EZ 0,97	Overlap	2000	3,5	4				X	3	Gap 0,1
65	PP2W65	EZ 0,97	EZ 0,97	Overlap	2000	3,5	4				X	3	Gap 0,1
66	PP2W66	EZ 0,97	EZ 0,97	Overlap	2000	3,5	4				X	2	Gap 0,1
67	PP2W67	EZ 0,97	EZ 0,97	Overlap	2000	3,5	4				X	3	Gap 0,1
68	PP2W68	EZ 0,97	EZ 0,97	Overlap	2000	3,5	4					1	Gap 0,1 / No welding, caused by irregular assembly pieces
69	PP2W69	EZ 0,97	EZ 0,97	Overlap	2000	3,5	2			X	X	2	Gap 0,1
70	PP2W70	EZ 0,97	EZ 0,97	Overlap	2000	3,5	2					1	Gap 0,1 / No welding, caused by irregular assembly pieces
71	PP2W71	EZ 0,97	EZ 0,97	Overlap	2000	3,5	2					3	Gap 0,1
72	PP2W72	EZ 0,97	EZ 0,97	Overlap	2000	3,5	2					1	Gap 0,1 / No welding, caused by irregular assembly pieces
73	PP2W73	EZ 0,97	EZ 0,97	Overlap	2000	3,5	2					1	Gap 0,1 / No welding, caused by irregular assembly pieces
74	PP2W74	EZ 0,97	EZ 0,97	Overlap	2000	3,5	-2					1	Gap 0,1 / No welding, caused by irregular assembly pieces
75	PP2W75	EZ 0,97	EZ 0,97	Overlap	2000	3,5	-2					1	Gap 0,1 / No welding, caused by irregular assembly pieces
76	PP2W76	EZ 0,97	EZ 0,97	Overlap	2000	3,5	-2					1	Gap 0,1 / No welding, caused by irregular assembly pieces
77	PP2W77	EZ 0,97	EZ 0,97	Overlap	2000	3,5	-2					1	Gap 0,1 / No welding, caused by irregular assembly pieces
78	PP2W78	EZ 0,97	EZ 0,97	Overlap	2000	3,5	-2					1	Gap 0,1 / No welding, caused by irregular assembly pieces
79	PP2W79	EZ 0,97	EZ 0,97	Overlap	2000	3,5	0					1	Gap 0,1 / Not welded
80	PP2W80	EZ 0,97	EZ 0,97	Overlap	2000	3,5	0					4	Gap 0,1
81	PP2W81	EZ 0,97	EZ 0,97	Overlap	2000	3,5	-4					3	Gap 0,1
82	PP2W82	EZ 0,97	EZ 0,97	Overlap	2000	3,5	-4					3	Gap 0,1
83	PP2W83	EZ 0,97	EZ 0,97	Overlap	2000	3,5	-4				V	3	Gap 0,1

Figure 6.3: P77 Datasheet III.

PARAMETER SHIT	Activity :	**METALLIC WELDING**	n° business :		Analitic code :	H226	Page :	4/4	**LASER FIBRE YLR-6000**	IRÉPALASER		
Project :	CCLW	Date of work :	17-18/06/2014	Opérator :	JR / JW	Work station :	Varilas 1	Foc :	*250*			
Fiber:	☑200 µm □300 µm □400 µm □600 µm	Type of nozzle :	□COAXIAL □OFF AXIS	Diameter of nozzle :	/	Beam incident angle :	90°	Reco :	*125*	ref info :		

Réf Shoot Réf précit.	**Stream n°**	**MATERIAL** mat. 1	mat. 2	Configuration	**CONSTANT PARAMETERS** P *in W*	— Speed *in m/min*	Location/upper surface Foc	**ERRORS** Spletter	Holes	undercut	Lack of pénétation	**EVALUATION** Error catégorie	**OBSERVATIONS** Unless specification, all welding are made with a 0,1mm gap comming from intersticial tinsel sheet and with previously cleaned sheets,
84	PP2W84	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	-4					1	Gap 0,1 / Not Welded
85	PP2W85	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	-4				X	3	Gap 0,1
86	PP2W86	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	-4				X	3	Gap 0,1 / Undercut : 197µm
87	PP2W87	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	-2					1	Gap 0,1 / Not Welded
88	PP2W88	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	-2					3	Gap 0,1
89	PP2W89	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	-2					3	Gap 0,1
90	PP2W90	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	-2			X		3	Gap 0,1
91	PP2W91	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	-2			X		3	Gap 0,1 / Undercut : 280µm
92	PP2W92	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0				X	4 until - 3 between- 3 after the last failure / 1: on the failures	Gap 0,1 / 2X SCOTCH (80-140 mm)
93	PP2W93	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0				X	1	Gap 0,1 / 2X SCOTCH (80-140 mm) / Not welded
94	PP2W94	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0				X	1	Gap 0,1 / 2X SCOTCH (80-140 mm) / Not welded
95	PP2W95	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0			X	X	2 until - 3 between- 3 after the last failure / 1: on the failures	Gap 0,1 / 2X SCOTCH (80-140 mm)
96	PP2W96	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0				X	1	Gap 0,1 / 2X SCOTCH (80-140 mm) / Not welded
97	PP2W97	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0				X	2 until the first failure / 1 after the failures	Gap 0,1 / 2 x Fat (80mm - 140mm from begining) / Not welded
98	PP2W98	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0				X	1	Gap 0,1 / 2 x Fat (80 mm - 140mm from begining) / Not welded
99	PP2W99	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0				X	2 until the first failure / 1 after the failures	Gap 0,1 / 2 x Fat (80mm - 140mm from begining) / Not welded
100	PP2W100	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0	X	X		X	2 until the first failure / 1 after the failures	Gap 0,1 / 2 x Fat (80 mm - 140mm from begining)
101	PP2W101	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0	X	X		X	4 until - 3 between- 3 after the last failure / 1: on the failures	Gap 0,1 / 2 x Fat (80 mm - 140mm from begining)
102	PP2W102	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0				X	1	Gap 0,1 / Without previous cleaning / Not welded
103	PP2W103	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0			X	X	1	Gap 0,1 / Without previous cleaning / Quite emergent
104	PP2W104	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0			X	X	1	Gap 0,1 / Without previous cleaning / Quite emergent (<30%)
105	PP2W105	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0		X	X	X	1	Gap 0,1 / Without previous cleaning / Quite emergent
106	PP2W106	EZ 0,97	EZ 0,97	Overlap welding	2000	3,5	0				X	1	Gap 0,1 / Without previous cleaning / Quite emergent

Figure 6.4: P77 Datasheet IV.



Quality scale	Data references
4 (perfect)	undercuts < 20% of the thickness from one metal sheet totaly emergent no spleters no holes and cuts
3 (good)	20% < undercuts < 30% emergent > 50% from the total welding length some spletters no holes and cuts
2 (acceptable)	30% < undercuts < 50% emergent < 50% from the total welding length some spletters holes and cuts < 20%
1 (unaceptable)	undercuts > 50% not emergent lots of spletters holes and cuts > 20%

Figure 6.5: P77 Datasheet V.

B - Autoencoder results for MNIST data set

To ensure the autoencoder results to be not significantly influenced by the inherent dependencies in the laser welding data set, the final stacked convolutional autoencoder has also been applied to the MNIST data set. All hyper parameters were kept the same as for the laser welding experiments, but only the data set was exchanged. In Figure 6.6 the filter reactions to a sample of the MNIST data set can be seen, as it passes through the layers. The fully connected layers are not shown as their representation is not easily interpretable in a spatial way. The convolutional filters behave similar to what has been seen for the laser welding data, indicating that the performance is not mainly dependent on the dependencies, inherent to the laser welding data. Figure 6.7 depicts the invariance

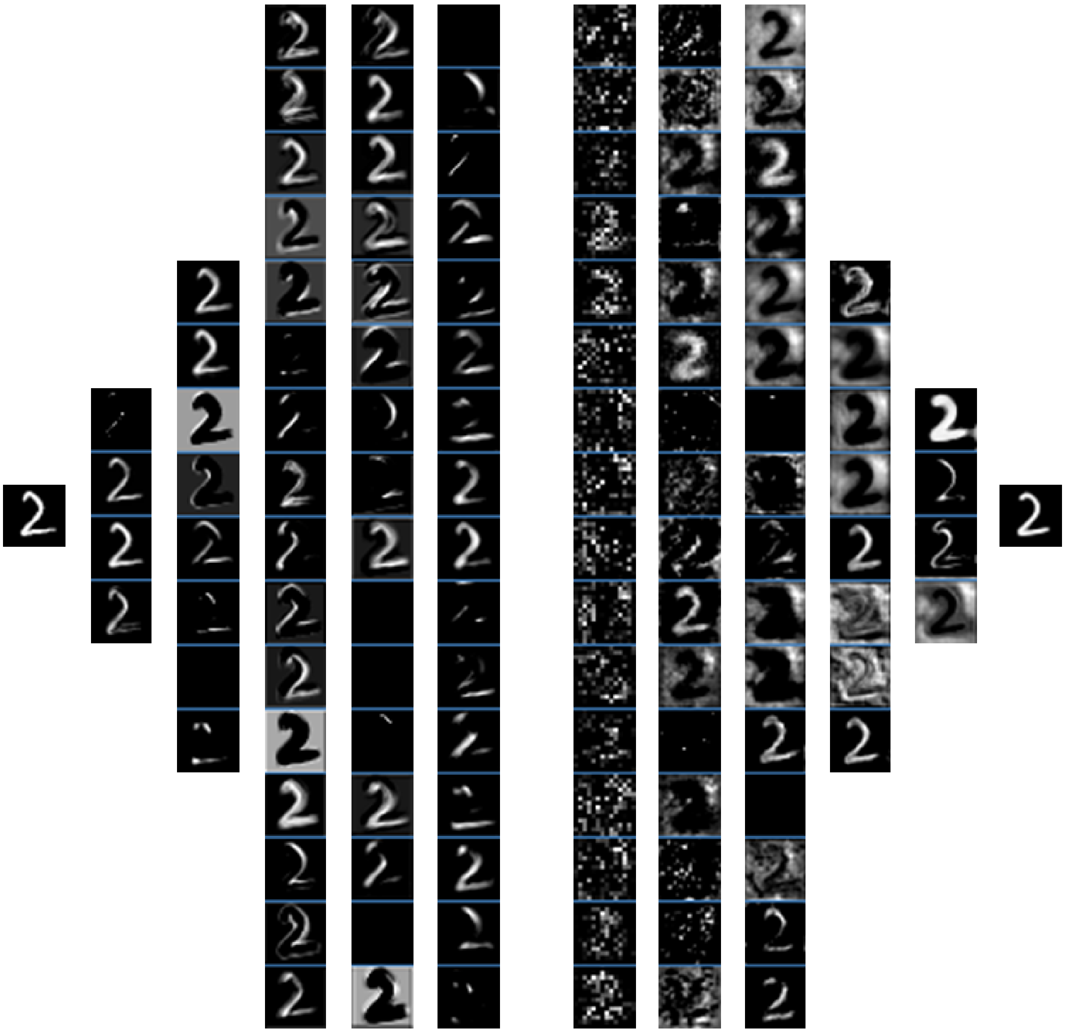
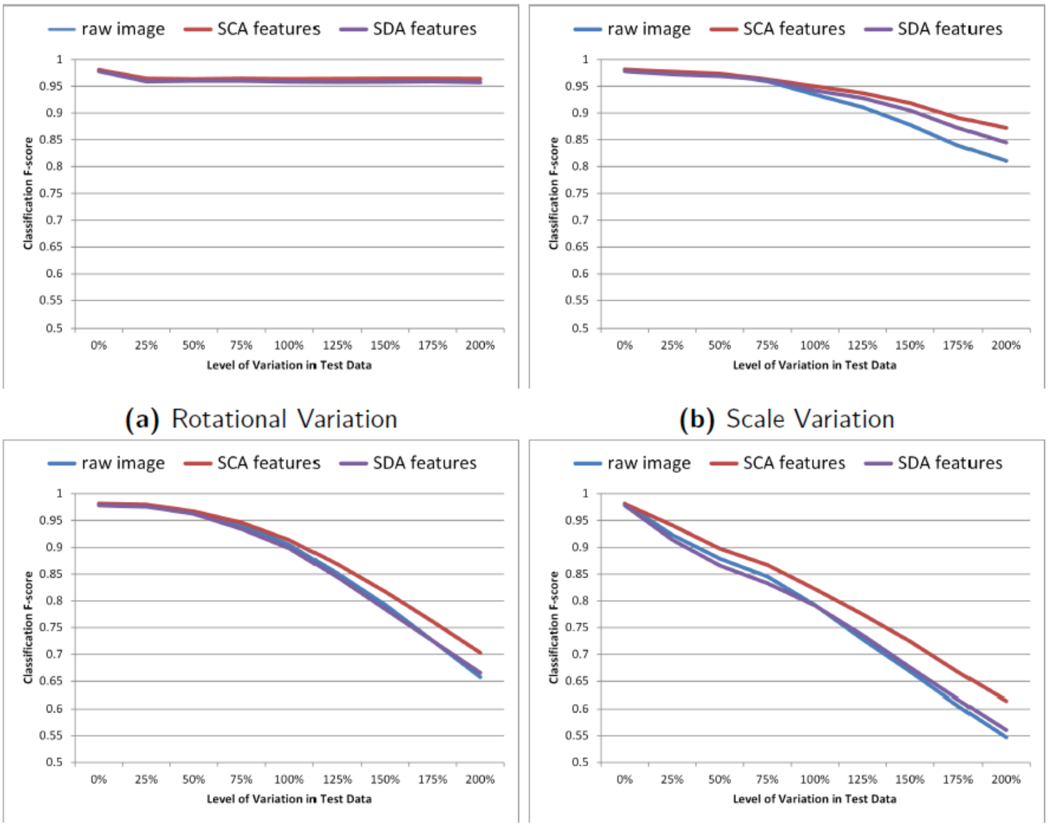


Figure 6.6: Visualization of a MNIST images, processed by the convolutional stacked autoencoder.

of the autoencoder extracted features from the MNIST data set for different kinds of variance. Similar to the laser welding data, the convolutional autoencoder performs best for all different types of variance. As seen for the other data set, rotation seems to have the least influence, while a combination of all three types of variance results in the worst classification performance. Nevertheless, these results proofs that the invariance, achieved by the autoencoding approaches are not predominantly dependent on the data set.



(a) Rotational Variation

(b) Scale Variation

(c) Translational Variation

(d) All three types of Variation

Figure 6.7: Classification performance, dependent on variance within the mnist images.