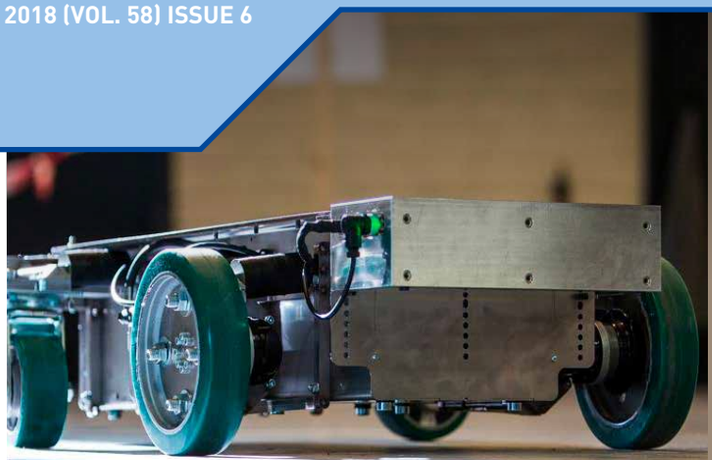
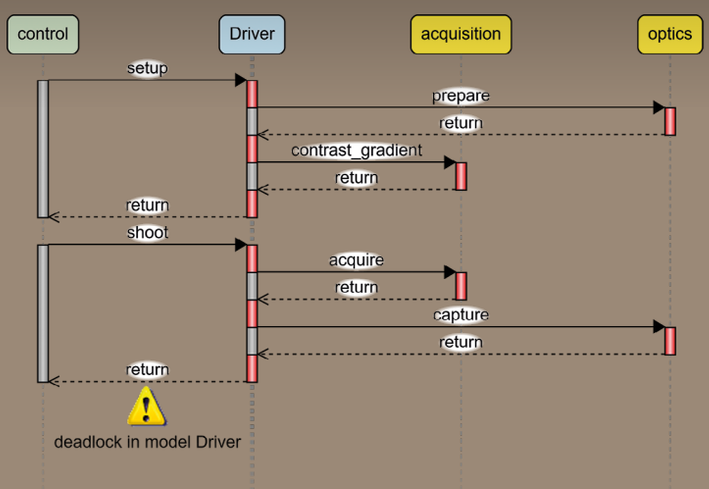
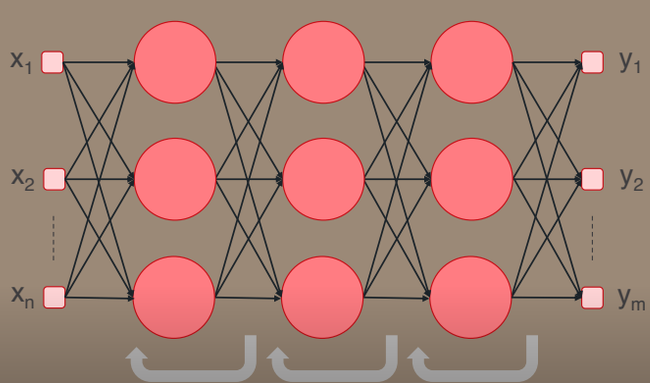
MIKRONIEK

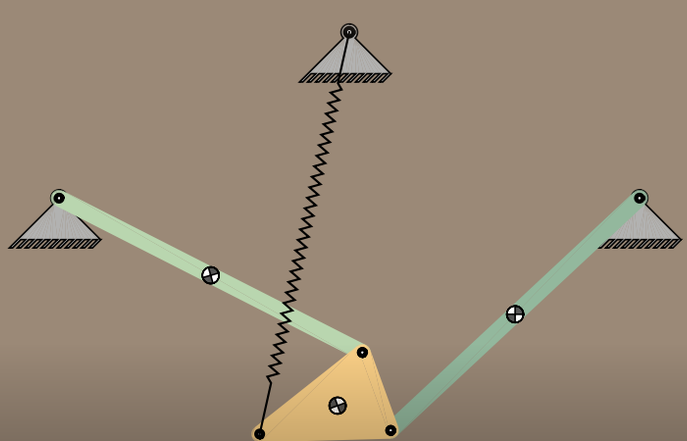
PROFESSIONAL JOURNAL ON PRECISION ENGINEERING

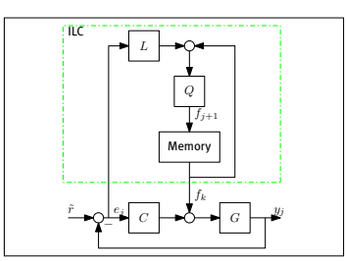
2018 (VOL. 58) ISSUE 6

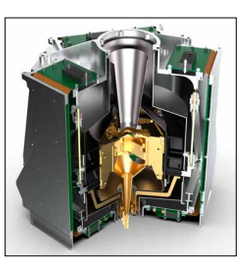


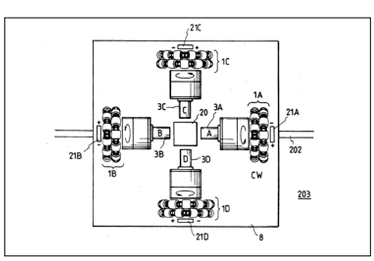












- THEME: SOFTWARE & MACHINE LEARNING
- STABLE X-RAY VISION FOR ATHENA SPACE TELESCOPE
- 2018 PRECISION FAIR IMPRESSIONS
- VALUABLE PATENT INFORMATION

DSPE

MIKRONIEK IS A PUBLICATION OF THE DUTCH SOCIETY FOR PRECISION ENGINEERING WWW.DSPE.NL

PUBLICATION INFORMATION

Objective
Professional journal on precision engineering and the official organ of DSPE, the Dutch Society for Precision Engineering. Mikroniek provides current information about scientific, technical and business developments in the fields of precision engineering, mechatronics and optics. The journal is read by researchers and professionals in charge of the development and realisation of advanced precision machinery.

DSPE
YOUR PRECISION PORTAL

Publisher
DSPE
Annemarie Schrauwen
High Tech Campus 1, 5656 AE Eindhoven
PO Box 80036, 5600 JW Eindhoven
info@dspe.nl, www.dspe.nl

Editorial board
Prof.dr.ir. Just Herder (chairman, Delft University of Technology, University of Twente),
Servaas Bank (VDL ETG), B.Sc.,
ir.ing. Bert Brals (Sioux Mechatronics),
dr.ir. Dannis Brouwer (University of Twente),
Maarten Dekker, M.Sc. (Philips),
Otte Haitsma, M.Sc. (Demcon),
ing. Ronald Lamers, M.Sc. (Thermo Fisher Scientific),
Erik Manders, M.Sc. (Philips Innovation Services),
dr.ir. Pieter Nuij (NTS-Group),
dr.ir. Gerrit Oosterhuis (VDL ETG),
Maurice Teuwen, M.Sc. (Janssen Precision Engineering)

Editor
Hans van Eerden, hans.vaneerden@dspe.nl

Advertising canvasser
Gerrit Kulsdom, Sales & Services
+31 (0)229 – 211 211, gerrit@salesandservices.nl

Design and realisation
Drukkerij Snep, Eindhoven
+31 (0)40 – 251 99 29, info@snep.nl

Subscription
Mikroniek is for DSPE members only.
DSPE membership is open to institutes, companies, self-employed professionals and private persons, and starts at € 80.00 (excl. VAT) per year.

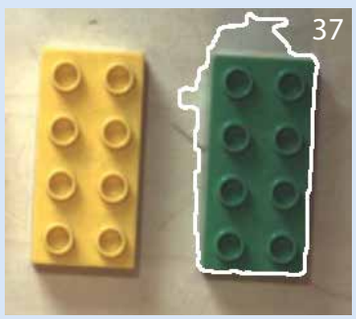
Mikroniek appears six times a year.

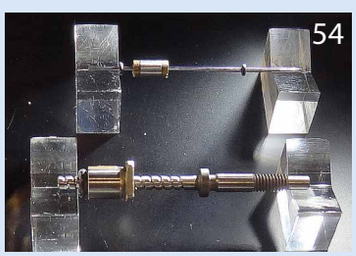
ISSN 0026-3699

The main cover photo collage (starting top left, clockwise) is courtesy of Nobleo Technology, Verum, Reinier Kuppens and Albert van Breemen. Read the articles on pages 34 ff, 28 ff, 24 ff and 12 ff, respectively.

IN THIS ISSUE

THEME: SOFTWARE & MACHINE LEARNING





FEATURES

INNOVATION AND DIGITALISATION OF SOFTWARE DEVELOPMENT

It takes software to turn the hardware of a complex high-tech machine into a dynamically operating system. Software is therefore an essential part of these systems, but it is upsetting to see how difficult it still is to develop that software without errors, and to prepare the code for a flawless integration into such a machine. In the 20 years that I have been visiting clients for my software company, I have observed very little change. There is a big gap between, for example, the mechanical engineers, who are busy with the physical system, and the software people. The latter think in code, while hardware people have much more of a system-level way of thinking. It is for this reason that we have stopped only executing software projects with our company and have switched to focus on the development of software products improving this.

Software development has to be done using a platform with which you can describe the desired behaviour of the system to be designed in a graphical, model-based way that is automatically converted to code. The hardware people are also able to work with this model; it creates a common language that bridges the gap between the software and hardware disciplines, allowing them to collaborate on the design at a system level.
Good riddance to manual, error-sensitive coding. The best way to create error-free code is to generate software automatically. What remains is errors at the design level. The more disciplines involved in software design (thanks to that common language), the more people can review and remove design errors. The next step is simulation: linking a model of the physical system to the 'real' operating software and then testing all sorts of scenarios to check whether the behaviour of the system is what the designers had in mind. This should mean that there are now very few mistakes left, and they will come to the fore when testing the physical system.

This method of software development and testing becomes all the more important if the design of a system, the hardware and the software, is regularly upgraded. This often takes place on the basis of machine learning: from the data generated by a system during its operation, improvements can be derived. Machine learning itself requires, for example, data analysis, and hence a lot of software – which also needs to be error free. Just like those updates themselves, of course. If, for example, the update of a smartphone does not work flawlessly, the market will not pick it up.
In short, software's share in high-tech projects is constantly increasing – partly under the influence of the digitalisation that Smart Industry brings with it – and innovation in the world of software is therefore urgently needed, otherwise the problem of errors will become totally uncontrollable. Digitalisation of software development (code generation, testing, simulation) should contribute to this. In the Dutch high-tech industry, however, the urgency of this has not yet been fully realised. This has to do with the fact that higher management of high-tech companies is mainly populated by hardware people.

Here lies a role for the High Tech Software Cluster, which in the Brainport region unites over 20 high-tech software companies in the areas of virtual prototyping & design, model-based software and data analytics & services. The ambition of this cluster is to contribute to shortening time-to-market and helping to prevent complex development projects become unverifiable and uncontrollable. A highly relevant cluster initiative is the Smart Industry fieldlab Software Competence Centre, which will work on innovation in software and on software-driven innovation, including topics such as digital twinning and model-based engineering.
When I see the massive commitment for the digitalisation of the industry in Germany, through the implementation of Industrie 4.0, I see also that we must take a firm stance in the Netherlands and embrace innovation and digitalisation of software software development. It's not yet too late.

Benno Beuting
CEO Cordis Automation
benno.beuting@cordis.nl, www.cordis.nl

LEARNING IN MACHINES

Control of high-tech mechatronic systems traditionally involves feedback and feedforward control, and essentially only uses a few recent measurements. Here, we aim to explore what can be learned from all available sensor data. A general learning framework is developed that exploits the abundance of data of previously executed tasks. Both fundamental insight and experimental results show that such iterative learning control approaches enable substantial performance improvement compared to traditional control. Interestingly, traditional model-based control theory turns out to have an essential role for fast and safe learning from measured data.

TOM OOMEN

Introduction

The learning from data and information has led to impressive achievements in recent years. Computer algorithms are now capable to successfully learn in many domains, including human language, ranging from speech recognition to accurate translations, real-time pattern recognition from images, digital advertising, self-driving vehicles, Atari, and Go [1]. The key enabler has been the availability of large amounts of data as well as ubiquitous and scalable computation and software.

In sharp contrast, high-tech mechatronic systems, such as manufacturing machines and scientific instruments, are often produced and installed with a pre-defined feedforward/feedback control algorithm, and their performance deteriorates over time due to wear, ageing and varying environmental conditions such as temperature variations. Examples range from lithography machines, 2D and 3D printers and pick & place robots, to microscopes and CT scanners.

Interestingly, these high-tech machines are prime examples of mechatronic system design, where control algorithms are typically implemented in a computer environment. Hence, over the lifetime of these high-tech machines, an abundance of data becomes available, yet this is often not exploited to enhance its performance. Indeed, sensors in mechatronic systems are often used for feedback control, which typically only makes use of real-time position and velocity information.

The aim of this article is to explore opportunities for learning from data in machines, possibly from past and already completed tasks, to control them to the limit of their physical capabilities. A framework for fast and safe learning is presented. Furthermore, at the end of the article, several practically relevant questions are addressed, including what should be done for a broad industrial deployment, what performance can be expected for a specific system at hand, and whether learning control can replace traditional feedback controllers.

Learning requirements

Learning in machines imposes several unique requirements, resulting from the fact that such machines are cyber-physical systems, involving interactions with the real world. In particular, the following requirements are considered throughout:

1. Learning should be fast, since machines require experiments in real-time. In addition, fast adaptation can be useful in case of varying operating conditions, e.g., due to temperature changes induced by motor heating or day/night periodicity.
2. Learning should be safe and use operational data, since dedicated experiments may induce production loss and even damage of the machine.

In the forthcoming sections, an approach to learning in machines is investigated that addresses these requirements.

Learning from past tasks

The aim of this section is to investigate the learning from data. This leads to an approach that bridges data-based learning and model-based control.

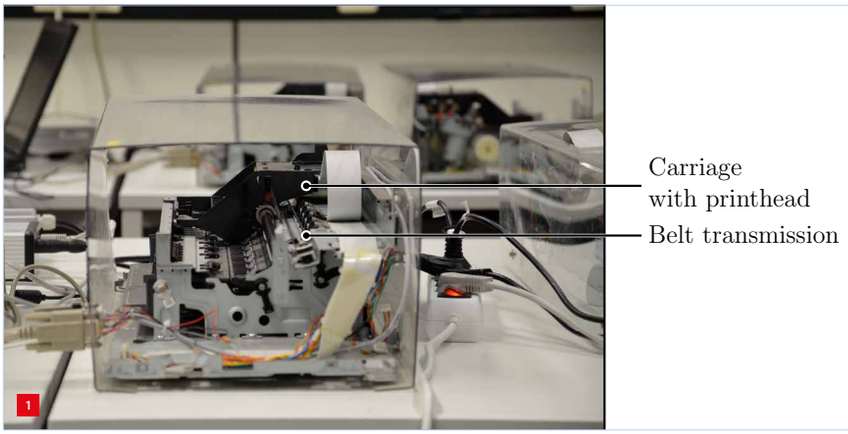
Traditional motion control

The printer in Figure 1 is considered as a key example of a mechatronic system. Here, the goal is to position the carriage that contains the printheads. A motor delivers an input u, which moves the carriage using a belt. The output position of the carriage y is measured using a linear encoder. The printer itself is denoted G.

AUTHOR'S NOTE

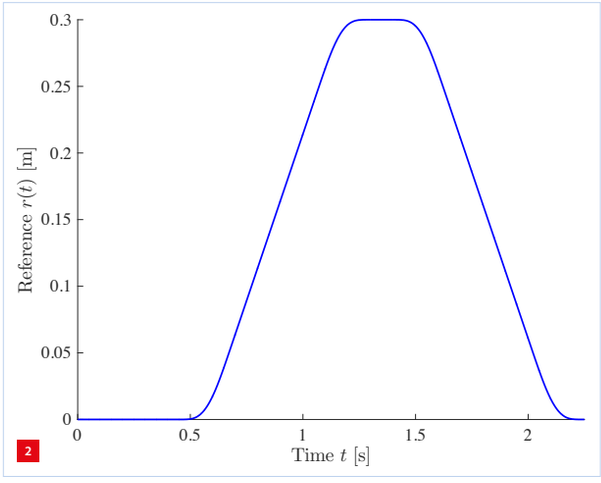
Tom Oomen is associate professor at Eindhoven University of Technology, the Netherlands, in the Control Systems Technology group at the department of Mechanical Engineering.

t.a.e.oomen@tue.nl
www.tue.nl/cst
www.toomen.eu

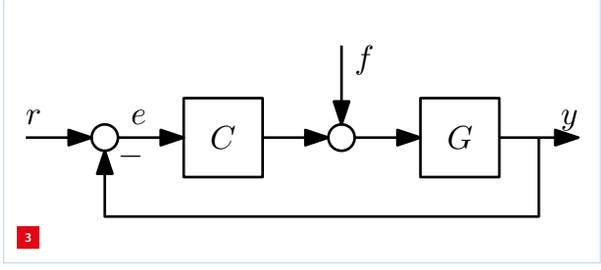


Printer system, used to illustrate traditional motion control and learning.

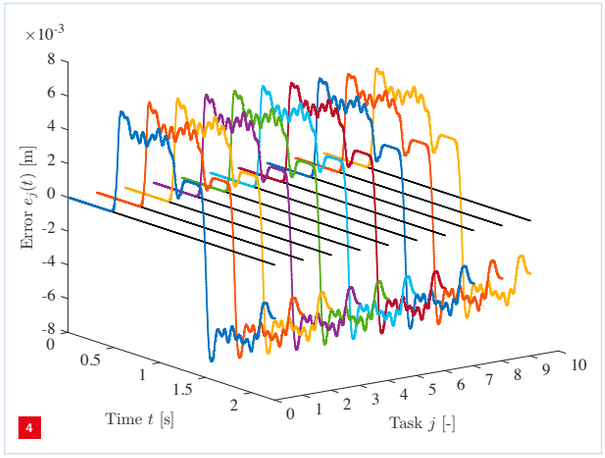
The control task is to track a reference trajectory r, such that the printhead moves over a sheet of paper, see Figure 2. The control problem is thus to choose the control input u such that the error $e = r - y$ is small. Traditionally, this is done using the controller structure shown in Figure 3. Here, C is a feedback controller. Feedforward control is implemented by selecting the signal f. A typical approach is to employ Newton's second law, $f = m \cdot a$, where m is an estimate of the mass and $a = \mathrm{d}^2 r/\mathrm{d}t^2$ is the acceleration profile.



Reference for the carriage containing the printheads.



Control architecture, where G denotes the system, C is the feedback controller, f is the feedforward input, r is the reference in Figure 2, and e is the tracking error.



Traditional motion control: the measured error signal is almost identical for subsequent tasks.

Motion control tasks are often performed repetitively. For example, the reference in Figure 2 has to be performed many times before a sheet of paper is printed: during each repetition of the reference in Figure 2, the sheet is moved a few millimeters by a sheet-positioning mechanism. The typical performance of traditional feedback motion control for such repetitive tasks is shown in Figure 4.

Here, ten tasks are shown, where in each task the reference in Figure 2 is tracked. The key observation is that the measured error is almost identical for each task j. Of course, feedforward control by selecting f can lead to a smaller error, but the key observation remains: the error is identical for each task, since the feedforward action f and feedback action Ce do not depend on past errors.

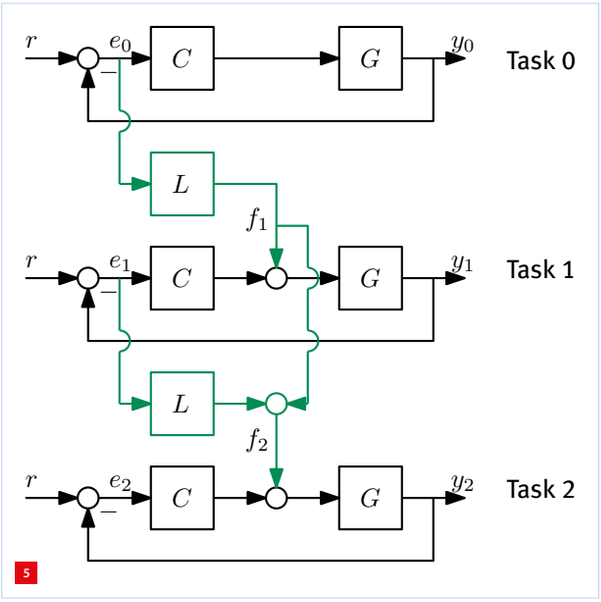
Learning from task to task

The observation that traditional motion controllers lead to a very similar error profile in Figure 4 raises the question: can we learn from past tasks, to improve the performance in the next task, i.e., task $j + 1$? Intuitively, the answer is affirmative: since the error is predictable, it can be compensated for. The practical question is how this can be achieved.

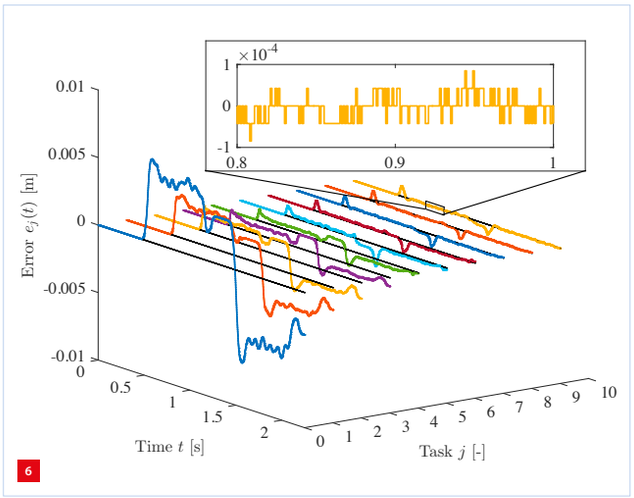
To learn from past tasks, assume that we perform the first task, $j = 0$, with no feedforward, thus $f_0 = 0$. The resulting error during the first task e_0 is then given by $e_0 = Sr$. Here, $S = 1/(1 + GC)$, the so-called sensitivity function, which can be directly derived from Figure 3. Now, consider the following idea. Assume that we measured e_0, but we do not have access to r. What feedforward f_1 should we select to reduce the error e_1? Note from Figure 5 that:

$$e_1 = Sr - GSf_1$$

Next, two key steps are made. First, note that we do not have direct access to Sr, but in fact it was measured in the



Towards learning from data of previous tasks.



Learning from past data in a printer system: fast convergence to encoder resolution.

earlier experiment: $e_0 = Sr$. Second, let f_1 depend on past errors, for instance:

$$f_1 = Le_0$$

Here, L is a design filter that still has to be specified, see also Figure 5. These steps directly lead to:

$$e_1 = e_0 - GSLe_0$$

This last equation immediately shows that the choice $L = (GS)^{-1}$ leads to $e_1 = 0$.

The update law $f_1 = Le_0$ with $L = (GS)^{-1}$ combines the data e_0 with model knowledge GS. Indeed, L is based on a model of the true closed-loop system GS. The key benefit of learning from data is that an approximate model suffices: of course we cannot expect to have access to an exact model of the system. If the model is not exact, then $GSL \neq 1$; so that e_1 is not zero, but typically much smaller than e_0. The central idea is to repeat the learning procedure in the next task $j = 2$:

$$f_2 = f_1 + Le_1$$

This essentially retains f_1 if it is perfect ($e_1 = 0$), and otherwise includes a small correction based on the already small e_1. This is then also done for future tasks:

$$f_{j+1} = f_j + Le_j$$

This idea of updating the control input is referred to as iterative learning control (ILC), see [2] for a historical overview.

Experimental results

Application of this procedure to the printer system in Figure 1 leads to the measured error signals in Figure 6. These results reveal impressive control performance: the error is at the level of the encoder resolution after only a few tasks. Hence, this very simple learning update leads to extremely high performance by combining data and model knowledge. Interestingly, these performance levels cannot be achieved using traditional feedforward and feedback controllers due to the presence of significant friction in the system; even though the learning update is a simple linear model it can perfectly compensate for these effects.

Can learning beat feedback?

Yes! The results in Figure 6 already reveal extremely high performance, which in practice cannot be achieved using traditional feedforward and feedback. The main reason is that feedback is subject to causality. This is well-known, since in $e_0 = Sr$, the term S cannot be made equal to zero due to the Bode Sensitivity Integral, often referred to by control engineers as the waterbed effect. The fundamental reason this integral exists is due to the fact that the physical system G is causal: it only responds to past outputs. In sharp contrast, in learning, one has access to what will happen in the (near) future due to the simple observation that this has been measured in past tasks. In practice, this is done by designing L to be a non-causal filter; practical details are provided in [3].

Fast and safe learning in the face of uncertainty

The role of model quality for learning

The results in Figure 6 reveal that the feedforward command signals that result from learning substantially increase control performance. In the previous section, it has been argued that the speed of learning depends on the

DIVING INTO DEEP LEARNING

Artificial intelligence (AI) technology has made rapid progress over the last few years. Breakthroughs in deep learning set new performance levels for various AI applications including speech recognition, language translation, recommendation and computer vision. It has enabled a first wave of successful consumer applications driven by companies like Google, Uber, Facebook and Netflix. Now a second AI wave is on the horizon, driven by industrial applications. This article gives an overview of the latest AI developments, the main techniques and their application to industrial and control engineering problems.

ALBERT VAN BREEMEN

Introduction

Applying AI to industrial and control engineering problems is not new. During the 70's and 80's expert systems became a popular AI technology to create industrial planning and diagnostics systems [1]. These systems take human expert knowledge and turn it into a database of if-then rules. The control software consists of a database with expert rules and some logic to process these rules ('inference logic'). This approach is particularly successful in domains where the problem cannot be modelled well using first principles and one has to rely on the intuition and knowledge of human experts.

During the early 90's a variation of expert systems using fuzzy logic became popular (see the box). The foundation of fuzzy set theory was laid in 1965 by Lofti Zadeh [1], but only during the late 80's it started to be applied to control engineering problems.

While rule-based AI systems provide a way to translate human expert knowledge into a (control) program, it is at the same time also limited by the knowledge of the human expert. During the mid-90's another AI technology became popular that tackled this problem. This AI technology is named artificial neural networks or just neural networks in short. A neural network is a machine learning AI algorithm that learns nonlinear relationships from data (Figure 1). It can be used as a generic building block to build non-linear adaptive control systems [3]. Applications of neural network-based control systems include nonlinear feedforward control in mechatronic positioning systems and optimised setpoint control [4].

The last few years a new AI technology is rapidly maturing. This technology is called deep learning and it builds upon the foundation of neural networks. The remainder of this article explains this technology and investigates its application to manufacturing and control problems.

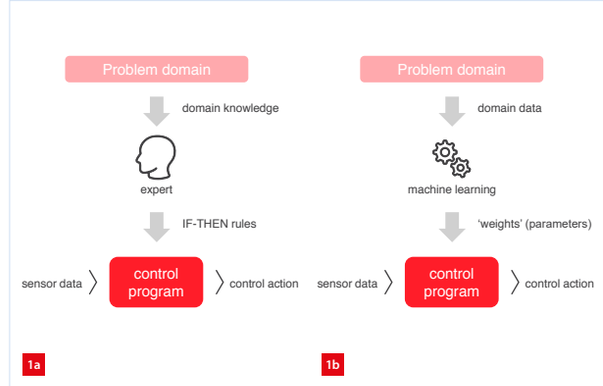
AUTHOR'S NOTE

Albert van Breemen is an independent AI & machine learning consultant with over 20 years of experience in high-tech technology and innovation.
He worked for companies including Philips and ASML.

info@aiblog.nl
www.aiblog.nl

Fuzzy logic

While expert systems are based on binary logic (yes, no), fuzzy logic uses truth values between 0 and 1. For example, a human expert might state a knowledge rule like IF temperate-is-high THEN set-heater-low. With binary logic the predicate temperate-is-high is defined by thresholding the temperature sensor data at a specific value. Fuzzy logic, however, defines a range of temperature values, each value being 'high', but with a different truth value. Fuzzy logic provides an intuitive method to design control systems for complex nonlinear processes that is robust to sensor data uncertainty. The technology was popular in Asia and several fuzzy logic-based controllers where used in consumer and industrial applications, such as rice cookers, washing machines and cement mills [2].



Expert-based design. *Machine learning-based design.*

Deep learning

Deep learning became popular after the 2012 edition of the yearly held 'ImageNet Large Scale Visual Recognition Challenge' (ILSVRC) [5]. During this challenge teams compete to develop the best program to detect objects in images. Up to 2012 the mainstream approach to build these programs was to use human expert knowledge and tuning. Alex Krizhevsky and his team, however, used a neural network with massively more parameters (62 million) than normally used.

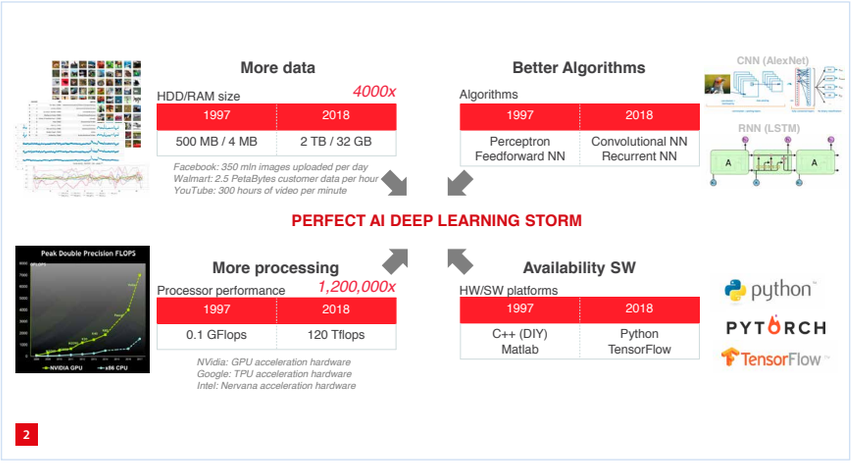
As training on a CPU processor would take too long, they used GPUs instead. A GPU has massively more computational cores than a CPU, with each core being able to perform basic multiply-and-addition operations of floating point numbers. As this is the core calculation of a neural network (see further), Krizhevsky et al. could speed up their training significantly.

In the end, they trained a winning neural network on two Nvidia GTX 580 GPUs (1,500 Gflops per GPU) in five to six days. Their solution achieved a 15.4% error rate, which was 10.8 percentage points better than the best system the previous year, and thus won the 2012 ILSVRC. Since then, other teams adopted this approach of using large neural networks, large datasets and GPU training. This approach is now called deep learning and has achieved impressive results in domains such as playing Go [7] and autonomous driving [8].

A perfect AI storm

Currently the field of AI is experiencing a perfect storm. Four technological developments accelerate the development of deep learning, since its conception in 2012 (Figure 2). These include:

1. Data storage & generation:
   Storage capacity in computer systems increased a factor of ~4,000 over the last 20 years. Simultaneously, cheap sensors and Internet connectivity fuel the creation of massive amounts of data; big data. Currently, big datasets of any domain, such as medical, retail, engineering or manufacturing, are freely available on websites such as *Kaggle.com* and *openml.org*, or are present within companies. Deep learning requires large datasets to train models and there is no technology bottleneck anymore for gathering and storing them.
2. Computational power:
   Processor power increased a factor of 1,200,000 over the last 20 years. Companies such as Nvidia and Google have been developing AI-optimised processors such as GPUs (graphical processing unit) and TPUs (tensor processing unit) [9]. These AI optimised processors give a performance boost of 10 to 50 times compared to CPUs.
3. AI algorithms:
   New types of neural networks and machine learning methods have been developed that more efficiently handle large amounts of data. The next section discusses the details of some of those algorithms.
4. Open-source software:
   The availability of open-source software has dramatically shortened the time to develop AI applications. Nowadays anybody has access to major AI software platforms, such as Python/Scikit-Learn [10], TensorFlow [11] or PyTorch [12]. Experimental results can be more easily shared and compared, which speeds up developments.



Perfect AI deep learning storm due to four technological developments.

Deep learning

Deep learning can be classified into four different types of algorithms, each optimised to handle specific data types and problems.

Artificial neural networks

The main technique behind deep learning is artificial neural networks (ANNs). An ANN is a model that is able to learn nonlinear relations between inputs and outputs. The basic building block of an ANN is a node called perceptron, which is a mathematical model of a biological neuron (see the box on the next page) and was formulated in 1943 by McCulloch & Pitts (referenced in [13]).

Mathematically, a perceptron is defined by the following equation:

$$= f(\sum_{i=1}^{n} w_i \cdot x_i + b)$$

with:

$$f(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}$$

Its output y is a function f of the summation of weighted inputs x_i and a bias. The parameters w_i and b (called 'weights' and 'bias') are determined by using a dataset and