IntechOpen

Deep Learning Applications

Edited by Pier Luigi Mazzeo and Paolo Spagnolo

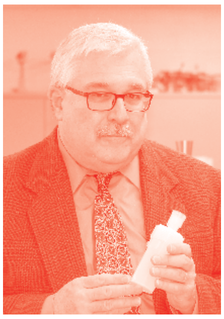
Deep Learning Applications

Edited by Pier Luigi Mazzeo and Paolo Spagnolo

Published in London, United Kingdom

IntechOpen

Supporting open minds since 2005



Deep Learning Applications
http://dx.doi.org/10.5772/intechopen.91576
Edited by Pier Luigi Mazzeo and Paolo Spagnolo

Contributors
Gonzalo Astray, Cecilia Martínez Castillo, Juan C. Mejuto, Manuel Alonso Ferrer, Wesam Salah Alaloul, Abdul Hannan Qureshi, Marina Ivasic-Kos, Kristina Hos, Miran Pobar, Hayrettin Okut, Talbi Mourad, Baazaoui Riadh, Bouhlel Med Salim, Marko Kesti, Stelios C. A. Thomopoulos, Cheng Huang, Liang Ma, Zhong-Sheng Liu, Rita Maria De Brito Alves, Fabio Machado Cavalcanti, Kelvin André Pacheco, Camila Emilia Kozonoe, Ayachi Errachdi, Sabrine Slama, Mohamed Benrejeb

First published in London, United Kingdom, 2021 by IntechOpen
IntechOpen is the global imprint of INTECHOPEN LIMITED, registered in England and Wales, registration number: 11086078, 5 Princes Gate Court, London, SW7 2QJ, United Kingdom
Printed in Croatia

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

Additional hard and PDF copies can be obtained from orders@intechopen.com

Deep Learning Applications
Edited by Pier Luigi Mazzeo and Paolo Spagnolo
p. cm.
Print ISBN 978-1-83962-374-5
Online ISBN 978-1-83962-375-2
eBook (PDF) ISBN 978-1-83962-376-9

Meet the editors

Pier Luigi Mazzeo obtained an MSc in Computer Science from the University of Salento, Lecce, Italy, in 2001. Since then, he has been working on several research topics regarding artificial intelligence and computer vision. Dr. Mazzeo joined the Italian National Research Council of Italy (CNR) as a researcher in 2002. He is currently involved in projects for algorithms for video object tracking, face detection and recognition, facial expression recognition, deep neural networks, and machine learning. He has authored and co-authored 100 publications, including more than fifteen papers published in international journals and book chapters. He has also co-authored five national and international patents. Dr. Mazzeo acts as a reviewer for several international journals and for some book publishers. He has been regularly invited to take part in the scientific committees of national and international conferences.

Paolo Spagnolo received an engineering degree in Computer Science from the University of Lecce, Italy, in 2002. Since then, he has been with the Italian National Research Council, working on several research topics regarding artificial intelligence and computer vision. He has studied techniques and methodologies for multidimensional digital signal processing, signal features extraction, supervised and unsupervised classification of signals, convolutional neural networks (CNN), and hyper- and multi-spectral imaging. He is the author of more than eighty papers on artificial intelligence. He also acts as a reviewer for several international journals. He has participated in several international projects in image and video analysis. He has been regularly invited to take part in the scientific committees of national and international conferences.

Contents

Preface

Deep learning is a branch of machine learning similar to artificial intelligence. As it is an exciting new field, many scientists and researchers are studying the applications of deep learning and proposing new architectures.

This book presents the state of the art, architectures, and systems of deep learning applications. The applications of deep learning vary from medical imaging to industrial quality checking, sports, and precision agriculture. The book is divided into two sections. The first section covers deep learning architectures, whereas the second section describes the state of the art of applications based on deep learning.

The editors thank the authors for their high-level contributions and their proactive collaboration in the realization of this book.

Pier Luigi Mazzeo and Paolo Spagnolo
National Research Council of Italy (CNR),
Institute of Applied Sciences and Intelligent Systems (ISASI),
Lecce, Italy

Section 1

Architectures

Chapter 1

Tuning Artificial Neural Network Controller Using Particle Swarm Optimization Technique for Nonlinear System

Sabrine Slama, Ayachi Errachdi and Mohamed Benrejeb

Abstract

This chapter proposes an optimization technique of Artificial Neural Network (ANN) controller, of single-input single-output time-varying discrete nonlinear system. A bio-inspired optimization technique, Particle Swarm Optimization (PSO), is proposed to be applied in ANN to avoid any possibilities from local extreme condition. Further, a PSO based neural network controller is also developed to be integrated with the designed system to control a nonlinear systems. The simulation results of an example of nonlinear system demonstrate the effectiveness of the proposed approach using Particle Swarm Optimization approach in terms of reduced oscillations compared to classical neural network optimization method. MATLAB was used as simulation tool.

Keywords: neural networks, particle swarm optimization, indirect control, nonlinear system

1. Introduction

We are interested, in this chapter, in adaptive system control of a class of single-input single-output (SISO) non-linear systems using neural network. In fact, this system control is a very general approach to adaptive control since one can combine in principle any parameter estimation scheme with any control strategy. In addition, its architecture is based on two neural network blocks corresponding to the system controller and the model identification of the dynamic behavior of the system [1, 2].

The use of artificial neural network (ANN) for identification, diagnosis, modeling and control has generated a lot of interest for quite some time now, because they have proved to be excellent function approximators, mapping any function to an arbitrary degree of accuracy, coupled with their ability for generalization, self-organization and self-learning [3].

Many architectures of neural networks are used. Among them, the most common and the most popular architecture is the multilayered perceptron, implemented with the standardized backpropagation algorithm. If the initial set of weights is not selected properly, this algorithm, employing a gradient descent search technique is seriously prone to getting trapped in local optimum solutions.

However, the calculations could slowly occurred and may even overflow or fluctuate between the optima. These limitations encouraged researchers to look for more powerful optimizations techniques that can help reach the optimal solution in an improved fashion, guarantee the convergence of control system and increase the learning speed [3].

A lot of optimization techniques of neural network are widely used. Among them, particle swarm optimization (PSO), the subject of this chapter, studied in different papers like in [3], and originates from the behavioral simulation of fish schooling and bird flocking. The conceptual model, at some point in the evolution of the algorithm, was an optimizer following which a number of parameters extraneous to optimization were eliminated leading to the basic PSO algorithm [3].

This technique is used in many applications of neural network optimization like identification, control and modeling. For instance, in [4], the authors used the PSO based neural network optimization for prediction of diameter errors in a boring machine. In their work, they established an improvement in the quality of optimization of the neural networks and error compensations with the use of the PSO algorithm, which achieves a better machining precision with fewer numbers of iterations.

The PSO algorithm is proposed to get optimal the parameters of ANN. This algorithm is well used because it has convergent result and not require many iterations, so in relative calculation relative quick. PSO is a population-based approach, which uses the swarm intelligence generated by the cooperation and competition between the particles in a swarm. It has been emerged successfully to a wide variety of search and optimization problems.

For example, in [5], the authors compared the performance of the PSO technique with other EAs for both continuous and discrete optimization problems in terms of processing time, convergence speed, and quality of the results. In addition, in [6], the authors proposed a PSO learning algorithm that self-generates radial basis function neural network (RBFN) to deal with three non-linear problems. This proposed PSO allows a high accuracy within a short training time when determining RBFN with small number of radial basis functions. Then, in [7], a PSO algorithm was developed by the authors to find the optimum process parameters which satisfy given limit, tool wear and surface roughness values and maximize the productivity at the same time. Also, in [8], the authors described an evolutionary algorithm for evolving the ANN which was based on the PSO technique. Both the architecture and the weights of the ANN were adaptively adjusted according to the quality of the neural network until the best architecture or a terminating criterion was reached. Moreover, the performance of the basic PSO algorithm with the constriction PSO on some test functions of different dimensions was compared by [9] and they found that the use of constriction PSO with mutation provided significant improvement in certain cases.

Further, in [10], it is presented an improved PSO algorithm for neural network training employing a population diversity method to avoid premature convergence. Furthermore, in [11], the authors used the PSO technique to optimize the grinding process parameters such as wheel and workpiece speed, depth and lead of dressing, etc. subjected to suitable constraints with the objective of minimizing the production cost and obtaining the finest possible surface finish. As well as, by comparing the PSO algorithm results with genetic algorithms and quadratic programming techniques, the PSO algorithm gives the global optimum solution with the objective to obtain minimum cost of manufacturing, [12]. Equally, in [13], the authors applied ANN - PSO approach for selection of optimum process parameters for minimizing burr size in drilling process. Besides that, the PSO algorithm was applied for optimization of multi objective problem in tile manufacturing process [14] and also for machinery fault detection [15]. Finally, in [16] the authors used PSO to tune the radial basis function networks for modeling of MIG welding process.

The PSO is well used in control system, for instance, in [17], the parameters of PID are tuned by ANN where their weights are optimized using PSO method to avoid any local minima/maxima in its searching procedure. In [18], the authors proposed a design of decentralized load-frequency controller for interconnected power systems with ac-dc parallel using PSO algorithm. The experiment result illustrated that their method have rapid dynamic response ability. In [19], the PSO algorithm is implemented to optimize the own five parameters of $PI^{\lambda}D^{\delta}$ controller via El-Khazali's approach in order to minimize several error functions satisfying some step response specifications such as the set of time domain and frequency domain constraints; overshoot, rise time and settling time. In [20], a comparative analysis of PSO algorithms is carried out, where two PSO algorithms, namely PSO with linearly decreasing inertia weight (LDW-PSO), and PSO algorithm with constriction factor approach (CFA-PSO), are independently tested for different PID structures.

In this chapter, a comparison of the performance of the PSO optimized neural network with the standard back-propagation is presented for the adaptive indirect control of nonlinear time-varying discrete system.

The present chapter is organized as follows. After this introduction, section 2 reviews the problem statement. Furthermore, in section 3 the neural network optimization methods are shown. In Section 4, tuning neural network controller using classical approach is presented. However, the section 5 details tuning neural network controller using PSO approach. An example of nonlinear system is studied, in section 6, to illustrate the proposed efficiency of the method. Section 7 gives the conclusion of this chapter.

2. Problem statement

The indirect adaptive control that is used, in this chapter, is composed of two blocks: a block of neural network model and a block of control system. The proposed control system is a neural network controller. At the simulation, it is assume that the neural network controller parameter's depending of the model parameter's as given in **Figure 1**.

In this architecture of indirect control, $r(k)$ is the desired value, $u(k)$ is the control law from the controller, $y(k)$ is the output of the nonlinear system, $yr(k)$ is the output of the neural network model, $e(k)$ is the identification error, $\hat{e}_c(k)$ is the estimated tracking error, $e_c(k)$ is the tracking error and k is the discrete time.

The aim of this chapter is to find a control law $u(k)$ to the nonlinear system, given by the Eq. (1), based on the tuning neural network controller's parameters in order that the system output $y(k)$ tracks, where possible, the desired value $r(k)$.

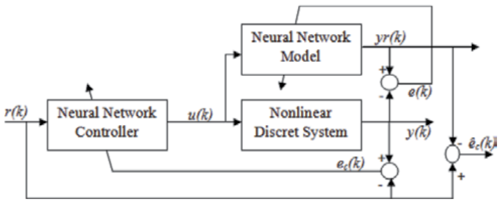


Figure 1.
The architecture of indirect neural control.

$$y(k+1) = f\left(y(k), \ldots, y\left(k-n_y\right), u(k), \ldots, u(k-n_u)\right) \tag{1}$$

$f(.)$ is the nonlinear function mapping specified by the model, n_y and n_u are the number of past output and input samples respectively required for prediction.

In this structure, the neural network controller and the neural network model must be updated at the same time. But, it is a difficult task to have a neural identifier learn the system fast enough to adapt to varying parameters. Therefore, the neural controller is ineffective on variations of the system parameters. In this chapter, to solve this problem we propose a fast learning algorithm based on a particle swarm optimization approach.

3. Neural network optimization methods

3.1 Gradient back-propagation algorithm

An Artificial Neural Networks (ANN) with randomly initialized weights usually shows poor results. That's why the most interesting characteristic of a neural network is its ability to learn, in other words, to adjust the weights of its connections according to the training data so that after training phase the faculty of generalization is obtained. This formulation turns the problem of learning into a problem of optimization.

In general, optimizing a system's parameters for a given task requires defining a metric that captures the inadequacy of the system for that task. This measure is called the cost function. Using an optimization algorithm, it is about finding the optimal parameters of the neural model that minimizes the cost.

For this kind of problem, there are two important classes of cost minimization search algorithms. Classical or gradient algorithms in which the central concept is that of direction of descent, are based on the derivatives of the cost function and any constraints, and advantageously made use of the specific information provided by the derivatives of different orders of these functions.

The alternative to these approaches is the use of meta-heuristics or heuristics such as genetic, stochastic, or evolutionary algorithms. Despite the notable advantage of not assuming regularity and their ability to locate the global minimum, these algorithms are strongly penalized by the relatively low convergence speeds and long computation times.

In the case of algorithms using the information provided by the derivatives of the functions defining the problem, each iteration comprises two main phases: the search for a direction of descent d_k and the determination of a step of descent η given by the following formula:

$$x_{k+1} = x_k + \eta\, d_k \tag{2}$$

The difference between these algorithms is manifested in the way these two steps are performed.

There are also three classes for such algorithms according to the strategy used to calculate the direction of descent:

1. gradient algorithms such as:

$$d_k = -\nabla f(x_k) \tag{3}$$

2. algorithms based on Newton's method in which the direction of descent is the solution of the system

$$\nabla^2 f(x_k) d_k = \nabla f(x_k) \tag{4}$$

3. the algorithms of the quasi-Newton type, in which an approximation H_k of the Hessian matrix evaluated in the iterates is built, the direction being then, as for the method of Newton, the solution of the linear system

$$H_k d_k = \nabla f(x_k) \tag{5}$$

The gradient back-propagation algorithm is the most widely used for weight adaptation, the goal of which is to find the appropriate combination of connection weights that minimizes the error function E defined by:

$$E = \frac{1}{2} \sum_k \left(y_k - yr_k\right)^2 \tag{6}$$

y_k and yr_k being, respectively, the desired output and the actual output of the k neuron for a given input vector.

This procedure is based on an extension of the Delta rule which involves a gradient descent and which consists in propagating an observation of the input of the neural network through the neural layer, to obtain the output values.

Compared to the desired outputs, the resulting errors allow the weights of the output neurons to be adjusted. Without the presence of the hidden layer, the knowledge of these errors allows a direct calculation of the gradient and makes the adjustment of the weights of these single neurons, easy as shown by the Delta rule. So for a network with hidden layers, ignoring the desired outputs of the hidden neurons, it thus remains impossible to know the errors of these neurons. So, as it is, this process cannot be used for weight adjustment of hidden neurons. The intuition which solves this difficulty and which gave rise to back-propagation was as follows: the activity of a neuron is linked to neurons of the preceding layer. Thus, the error of an output neuron is due to the hidden neurons of the previous layer in proportion to their influence; therefore according to their activation and the weights that connect the hidden neurons to the output neuron. Therefore, we seek to obtain the contributions of the L hidden neurons which gave the error of the output neuron k.

The back-propagation procedure consists in propagating the error gradient (error produced during the propagation of an input vector) in the network. In this phase, the propagation of an output neuron's error starts from the output layer to the hidden neurons.

It is therefore sufficient to retrace the original activation path backwards, starting from the errors of the output neurons, to obtain the error of all the neurons in the network. Once the corresponding error for each neuron is known, the weight adaptation relationships can be obtained.

3.2 Second order optimization method

Another class of methods, more sophisticated than the previous one, is based on second order algorithms, based on Newton's method which adapts the weights according to the following relation:

$$\Delta w = -H^{-1} \nabla E \tag{7}$$

where the element H_{ij} of the Hessian matrix H relates to the second partial derivatives of the cost function, compared to the weights. The elements of this matrix are defined by

$$H_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j} \tag{8}$$

Like gradient-only methods, second-order methods determine the gradient by the back-propagation algorithm and generally approximate the Hessian matrix or its inverse, since the cost of its computation may quickly become prohibitive.

This type of method localizes, in a single iteration, the minimum of a quadratic empirical error criterion and requires several iterations when this criterion is not ideally quadratic.

In practice, the convergence of the corresponding algorithm towards an optimal solution is rapid so that a good number of error hyper-surfaces present a quadratic curvature in the immediate vicinity of the minima. This method nevertheless remains subject, when the error hyper-surfaces are complex, to convergence towards non-minimal solution points for which the gradient of the empirical error criterion is canceled out at the inflection points or at the saddle points. In addition, there are the possibilities of divergence of the algorithm when the Hessian matrix is not positive definite.

The evaluation and memorization of the inverse Hessian matrix, on which the second-order methods are based, is, however, a major handicap in the context of learning large networks.

But, the main drawback lies in the calculation of the second derivatives of E which is most often expensive and very difficult to carry out. A certain number of algorithms propose to get around this difficulty by using approximations of the Hessian matrix.

This approximation, at the basis of Gauss-Newton or Levenberg–Marquardt algorithms, is widely used in the identification of rheological parameters. This method is adapted especially for the problems of small dimensions since the computation of the Hessian matrix is easy. Whereas if the problem presents a large number of variables, it is generally advised to couple it with the conjugate gradient method or a Quasi-Newton method. Or, when the relative improvement in the objective function becomes too low, we automatically switch to the conjugate gradient method.

The Levenberg–Marquardt method, Marquardt method, another second-order method, very close to the Newton method described previously, in fact offers an interesting alternative by adjusting the weights as following:

$$\Delta w_{ij} = -[H + \mu I]^{-1} \nabla E \tag{9}$$

μ is the Levenberg–Marquardt parameter and I is the identity matrix.

This method, making a compromise between the direction of the gradient and Newton's method, has the particularity of adapting to the shape of the error surface. Indeed, for low values of μ, the Levenberg–Marquardt method approaches Newton's method, and for large values of μ, the algorithm is quite simply a function of the gradient, knowing that the parameter μ is automatically updated based on the convergence of each iteration.

Stabilization is possible thanks to a reiterative process, i.e. if an iteration diverges, it can be started again by increasing the parameter μ until a convergent iteration is obtained. However, the phenomenon of strong divergence when approaching the optimum, inherent in Newton's method, is in no way suppressed here. At most, the divergence can be reduced.

Despite the interesting properties of this method, calculating the inverse of $(H + \mu I)$, makes its use tricky for heavy neural networks. As a result, like Newton's method, it is advisable to automatically switch to the conjugate gradient method when this divergence phenomenon appears. Second-order methods greatly reduce the number of iterations, but increase the computation time.

3.3 Heuristic optimization methods

The advantage of heuristic optimization methods is the minimization of non-derivable cost functions, even for a large number of parameters $\left(1 < n < 10^5\right)$.

Among the effective methods, we distinguish the optimization algorithm by Particle Swarms introduced by Kennedy and Eberhart and improved by Clerc is an optimization technique by agents which is essentially inspired by the behavior social in flocks of birds or schools of fish.

In addition, genetic algorithms, evolutionary algorithms, also constitute stochastic optimization techniques inspired by the theory of evolution according to Darwin, now widely used in numerical optimization, when the functions to be optimized are complex, irregular, poorly known or in question. Combinatorial optimization.

These heuristic methods of the methods presented previously (Levenberg–Marquardt, Newton, Conjugate Gradient, …) by three main aspects:

1. they do not require the gradient calculation,

2. they study a population as a whole while deterministic methods treat an individual who will evolve towards the optimum,

3. they involve random operations.

Experience has also shown that if the components as well as the evolution parameters are carefully tuned, it is possible to obtain extremely efficient and fast algorithms. However, this adjustment step can be very delicate and constitutes a drawback of the implementation of these methods.

4. Tuning neural network controller using classical approach

The architecture shown in **Figure 1** assumes the role of two neural blocks. Indeed, the weights of the neural model are adjusted by the identification error $e(k)$, however the weights of the neural controller are trained by the tracking error $e_c(k)$.

The multi-layer perceptron is used in the neural model and in the neural controller. Each block consists of three layers. The sigmoid activation function s is used for all neurons.

Concerning the neural network model, the j^{th} output layer of the hidden layer is described as follows

$$h_j = \sum_{i=1}^{n_1} w_{ji} x_i \quad j = 1, 2, \dots, n_2 \tag{10}$$

where n_1 is the number of nodes of the input layer, w_{ji} is the hidden weight, x_i is the input vector of the neural model, $x = [u(k), u(k-1), u(k-2), \dots]^T$, $u(k)$ is the control input to the system and n_2 is the number of nodes of the hidden layer given in the expression (3).

The output of the neural network model is given by the following equation

$$yr(k+1) = \lambda\, s\left(\sum_{j=1}^{n_2} w_{1j} s\left(h_j\right)\right) \tag{11}$$

where w_{1j} is the weight from the hidden layer to the output layer and λ is a scaling coefficient. The compact form of the output is given by the following equation

$$yr(k+1) = \lambda\, s(h_1) = \lambda\, s\left[w_1^T S(Wx)\right] \tag{12}$$

with

$$\begin{aligned} x &= [x_i]^T, i = 1, \dots, n_1, \\ W &= \left[w_{ji}\right], i = 1, \dots, n_1, j = 1, \dots, n_2, \\ S(Wx) &= \left[s\left(h_j\right)\right]^T, j = 1, \dots, n_2, \\ w_1 &= \left[w_{1j}\right]^T, j = 1, \dots, n_2. \end{aligned}$$

The incremental change of the hidden weights Δw_{ij}, $i = 1, \dots, n_1$ and $j = 1..n_2$, is

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} = -\eta \frac{\partial E}{\partial e} \frac{\partial e}{\partial h_1} \frac{\partial h_1}{\partial h_j} \frac{\partial h_j}{\partial w_{ji}} \tag{13}$$

$$\Delta w_{ji} = \eta \lambda s'(h_1) S'(Wx) w_{1j} x^T e(k) \tag{14}$$

with η is the learning rate, $0 \le \eta \le 1$, $S'(Wx) = diag\left[s'\left(h_j\right)\right]^T, j = 1, \dots, n_2$, $s'(h_1)$ is the derivative of $s(h_1)$ defined as follows

$$s'(h_1) = s(h_1)(1 - s(h_1)) \tag{15}$$

$e(k)$ is the identification error which is given by

$$e(k) = y(k) - yr(k) \tag{16}$$

and the function cost which is given by the following equation

$$E = \frac{1}{2}\sum_{k=1}^{N} (e(k))^2 = \frac{1}{2}\sum_{k=1}^{N} (y(k) - yr(k))^2 \tag{17}$$

where N is the number of observations.

The incremental change of the hidden weights Δw_{ij} is used in the following equation

$$w_{ji}(k+1) = w_{ji}(k) + \Delta\, w_{ji}(k) \tag{18}$$

However, the output weights are updated by the following equation

$$w_{1j}(k+1) = w_{1j}(k) + \Delta\, w_{1j}(k) \tag{19}$$

where Δw_{1j} is

$$\Delta w_{1j} = -\eta \frac{\partial E}{\partial w_{1j}} = -\eta \frac{\partial E}{\partial e} \frac{\partial e}{\partial h_1} \frac{\partial h_1}{\partial w_{1j}} \tag{20}$$

$$\Delta w_{1j}(k) = \eta \lambda e(k) s'(h_1) S(Wx) \tag{21}$$

Concerning the neural network controller, the j^{th} output layer of the hidden layer is

$$h_{cj} = \sum_{i=1}^{n_3} v_{ji} x_{1i} \quad j = 1, \dots, n_4 \tag{22}$$

where n_3 is the number of nodes of the input layer, v_{ji} is the hidden weight and x_{1i} is the input vector of the neural network controller $x_1 = [r(k), r(k-1), r(k-2), \dots]^T$, $r(k)$ is the desired value.

The output of the neural network controller is given by the following equation

$$u(k) = \lambda_c\, s\left(\sum_{j=1}^{n_4} v_{1j} s\left(h_{cj}\right)\right) = \lambda_c\, s\left(\sum_{j=1}^{n_4} v_{1j} s\left(\sum_{i=1}^{n_3} v_{ji} x_{1i}\right)\right) \tag{23}$$

where n_4 is the number of nodes of the hidden layer, λ_c is a scaling coefficient and v_{1j} is the output weight.

The compact form of the output of the neural network controller is given by the following equation

$$u(k) = \lambda_c\, s(h_{c1}) = \lambda_c\, s\left[v_1^T S(Vx_1)\right] \tag{24}$$

with

$$x_1 = [x_{1i}]^T, i = 1, \dots, n_3,$$
$$V = \left[v_{ji}\right], i = 1, \dots, n_3, j = 1, \dots, n_4,$$
$$S(Vx_1) = \left[s\left(h_j\right)\right]^T,\ j = 1, \dots, n_4,$$
$$v_1 = \left[v_{1j}\right]^T, j = 1, \dots, n_4.$$

Concerning the hidden synaptic weights, they are updated by

$$v_{ji}(k+1) = v_{ji}(k) + \Delta\, v_{ji}(k) \tag{25}$$

where Δv_{ji} is given by

$$\Delta v_{ji} = -\eta_c \frac{\partial E_c}{\partial e_c} \frac{\partial e_c}{\partial y} \frac{\partial yr}{\partial h_1} \frac{\partial h_1}{\partial s\left(h_j\right)} \frac{\partial s\left(h_j\right)}{\partial h_j} \frac{\partial h_j}{\partial u} \frac{\partial u}{\partial h_{c1}} \frac{\partial h_{c1}}{\partial h_{cj}} \frac{\partial h_{cj}}{\partial v_{ji}} \tag{26}$$

with η_c is the learning rate, $0 \le \eta_c \le 1$ and the function cost defined as follows

$$E_c = \frac{1}{2}\sum_{k=1}^{N}(e_c(k))^2 = \frac{1}{2}\sum_{k=1}^{N}(y(k) - r(k))^2 \tag{27}$$

where N is the number of observations and $e_c(k)$ is the tracking error which is given by the following equation

$$e_c(k) = y(k) - r(k) \tag{28}$$

where $r(k)$ is the desired output. So Δv_{ji} comes

$$\Delta v_{ji} = \eta_c \lambda_c e_c(k) s'(h_1) w_{1j} S'(Wx) w_{ji} s'(h_{c1}) v_{1j} S'(Vx_1) x_1^T \tag{29}$$

with $S'(Vx_1) = diag\left[s'\left(h_j\right)\right]^T, j = 1, \dots, n_4$.
The output synaptic weights of the neural network controller are updated as

$$v_{1j}(k+1) = v_{1j}(k) + \Delta\, v_{1j}(k) \tag{30}$$

where Δv_{1j} is given by

$$\Delta v_{1j} = -\eta_c \frac{\partial E_c}{\partial v_{1j}} = -\eta_c \frac{\partial E_c}{\partial e_c}\frac{\partial e_c}{\partial h_{c1}}\frac{\partial h_{c1}}{\partial v_{1j}} \tag{31}$$

So

$$\frac{\partial e_c(k)}{\partial h_{c1}} = \frac{\partial (y(k) - r(k))}{\partial h_{c1}} = \frac{\partial y(k)}{\partial h_{c1}} = \frac{\partial y(k)}{\partial u(k)}\frac{\partial u(k)}{\partial h_{c1}} \tag{32}$$

and the Eq. (31) becomes

$$\Delta v_{1j} = -\eta_c \frac{\partial E_c}{\partial e_c}\frac{\partial y(k)}{\partial u}\frac{\partial u(k)}{\partial h_{c1}}\frac{\partial h_{c1}}{\partial v_{1j}} \tag{33}$$

or in Eq. (33), $y(k)$ does not depend on h_1, for this reason we use $yr(k)$ instead of $y(k)$ under the condition that the neural model is equal to the system behavior which gives

$$\frac{\partial y(k)}{\partial u} = \frac{\partial yr(k)}{\partial u} = \frac{\partial yr(k)}{\partial h_1}\frac{\partial h_1}{\partial s\left(h_j\right)}\frac{\partial s\left(h_j\right)}{\partial h_j}\frac{\partial h_j}{\partial u} \tag{34}$$

from where approximately

$$\Delta v_{1j} = -\eta_c \frac{\partial E_c}{\partial e_c(k)}\frac{\partial e_c(k)}{\partial y(k)}\frac{\partial yr(k)}{\partial h_1}\frac{\partial h_1}{\partial s\left(h_j\right)}\frac{\partial s\left(h_j\right)}{\partial h_j}\frac{\partial h_j}{\partial u(k)}\frac{\partial u(k)}{\partial h_{c1}}\frac{\partial h_{c1}}{\partial v_{1j}} \tag{35}$$

the obtained incremental change Δv_{1j} is rewritten as

$$\Delta v_{1j} = \eta_c \lambda_c e_c(k) s'(h_1) w_{1j} S'(Wx) w_{ji} s'(h_{c1}) S(Vx_1) \tag{36}$$

In this section, we used a fixed learning rate, $\eta(k)$ (respectively $\eta_c(k)$), and a derivative of sigmoid function $s'(h_1) = s(h_1)(1 - s(h_1))$.

This approach has two drawbacks. First, to find a suitable fixed learning rate $\eta(k)$ (respectively $\eta_c(k)$), several tests are called which decreases the on-line operation. Second, when we use this type of derivative of sigmoid function, a large amount of error should not be spread to the weights of the output layer and the learning speed becomes very slow. In order to increase the learning speed, some of the new proposed approaches are proposed in the next section.

5. Tuning neural network controller using particle swarm optimization

An alternative technique is proposed, in this section, to optimize the neural network controller by implementing Particle Swarm Optimization algorithm. This algorithm works like animal behavior on finding foods and avoiding danger, where they will coordinate with each other to find the best position to settle. Likewise, PSO is directed by the movement of the best individual from the population, known as the social compound, and their own experience, known as the cognitive compound. The algorithm moves the set of solutions to find the best solution among them.

5.1 Mathematical formulation

In this study, the Particle Swarm Optimization Feedforward Neural Network (PSO NN) is applied to a multi-layered perceptron where the position of each particle, in a swarm, represents the set of synaptic weights of the neural network for the current iteration. The dimensionality of each particle is the number of synaptic weights.

Let us consider a search space of dimension D. A particle i of the swarm is modeled by a position vector

$$x_{ij} = [x_{i1},\ x_{i2},\ \dots, x_{iD}]^T \tag{37}$$

and a velocity vector denoted

$$v_{ij} = [v_{i1},\ v_{i2},\ \dots, v_{iD}]^T \tag{38}$$

There is no concept of backpropagation in PSO NN where the direct neural network produces the learning error, objective function of each particle, based on the set of synaptic weights and biases, the positions of the particles. Each particle moves in the weighting space trying to minimize the learning error and keeps in memory the best position through which it passed, denoted

$$Pbest_{ij} = \left[pbest_{i1},\ pbest_{i2},\ \dots, pbest_{iD}\right]^T \tag{39}$$

whereas the best position reached by the swarm is denoted

$$Gbest_{ij} = \left[gbest_{i1},\ gbest_{i2},\ \dots, gbest_{iD}\right]^T \tag{40}$$

Changing the position means updating the synaptic weights of the neural network controller to generate the proper control law by reducing tracking error.

In each iteration k, the particles update their position by calculating the new velocity and move to the new position. At the iteration $(k+1)$, the velocity vector is calculated as follows:

$$v_{ij}(k+1) = wv_{ij}(k) + c_1r_1\left[pbest_{ij}(k) - x_{ij}(k)\right] + c_2r_2\left[gbest_{ij}(k) - x_{ij}(k)\right] \tag{41}$$

where:

w being a variable parameter making it possible to control the changing of the particle at the next iteration,

wv_{ij} is a physical changing component,

$c_1r_1\left[pbest_{ij}(k) - x_{ij}(k)\right]$ is a cognitive changing component,

$c_2r_2\left[gbest_{ij}(k) - x_{ij}(k)\right]$ a social component of changing,

c_1 and social c_2 are respectively, two cognitive confidence coefficients and social which are present the degree of attraction towards the best position of a particle and that of these informants,

r_1 and r_2 are two random numbers drawn uniformly in the interval $[0, 1]$ represent the proper exploration of particles in the search space.

The smallest learning error of each particle $Pbest_i$ and the smallest learning error found in the whole learning process $Gbest_i$ are applied to produce a fit of the positions towards the best solution or the targeted tracking error.

The position, at the iteration $(k+1)$, of particle i is then defined as follows:

$$x_{ij}(k+1) = x_{ij}(k) + v_{ij}(k) \tag{42}$$

Once the change in positions has taken place, an update affects both $Pbest_i$ and $Gbest_i$ vectors. At the iteration $(k+1)$, these two vectors will be updated according to the following two formulations:

$$Pbest_i(k+1) = \begin{cases} Pbest_i(k) & if \quad f(x_i(k)) \geq f(Pbest_i(k)) \\ x_i(k) & else \end{cases} \tag{43}$$

and

$$Gbest_i(k+1) = Arg \min_i \left[Pbest_i(k+1)\right] \tag{44}$$

The algorithm is executed as long as one of the three, or all at the same time, of the following convergence criteria is verified:

- the maximum number of iterations defined has not been reached,
- the variation in particle speed is close to zero,
- the value of the objective is satisfactory, with respect to the following relation:

$$|f\left(gbest_{jk}(k)\right) - f\left(gbest_{jk}(k-\alpha)\right)| \leq \varepsilon, \quad with \;\; \alpha \in [1, N] \tag{45}$$

The parameter ε represents a tolerance chosen, most often, of the order of 10^{-5} and N is a number of iterations chosen of the order of 10.

5.2 The proposed algorithm of particle swarm optimization

In this section, a summary of the proposed algorithm of the PSO neural network controller is presented.

- Random initialization of the positions and velocity of the N particles in the swarm,
- **For** k: 1..N **Do**,
- Repeat,
- **For** all particles i **Do**,
- Calculation of the control law $u_i(k)$ from the controller input vector $x_c(k)$,
- Calculation of the outputs of the system $y_i(k+1)$,
- Evaluation of the positions of particles in the research space,
- **If** the current positions of particle i produce the best objective function in its history **Then**,
- $Pbest_i \leftarrow e_{c_i}$,
- **If** the objective function of particle i is the best overall objective function **Then,**
- $Gbest_i \leftarrow e_{c_i}$,
- End If,
- End If,
- End For,
- Moving of particles according to Eqs. (41) and (42),
- Evaluation of particle positions,
- Update $Pbest_i$ and $Gbest_i$ according to Eqs. (43) and (44),
- **Until** reaching the stop criterion,
- End of PSO.

In this section, we have proposed the PSO optimization steps of an indirect neural network adaptive controller. The corresponding algorithm will be applied for discrete SISO nonlinear systems.

6. Results and discussion

In this section, a time-varying nonlinear discrete systems is used which is described by the input–output model in the following Equation [21].

$$y(k+1) = \frac{y(k)y(k-1)y(k-2)u(k-1)(y(k-2)-1)+u(k)}{a_0(k)+a_1(k)y^2(k-1)+a_2(k)y^2(k-2)} \tag{46}$$

where $y(k)$ and $u(k)$ are respectively the output and the input of the time-varying nonlinear system at instant k; $a_0(k)$, $a_1(k)$ and $a_2(k)$ are given by

$$\begin{cases} a_0(k) = 1 \\ a_1(k) = 1 + 0.2\,cos\,(k) \\ a_2(k) = 1 + 0.2\,sin\,(k) \end{cases} \tag{47}$$

The trajectory of $a_1(k)$ and $a_2(k)$ are given in the following **Figure 2**.

In this section, in order to examine the effectiveness of the proposed algorithm of neural network controller and the PSO neural network controller different performance criteria are used. Indeed, the mean squared tracking error (MSE_{e_c}) and the mean absolute tracking error (MAE_{e_c}) are, respectively, given by respectively, given by

$$MSE_{e_c} = \frac{1}{N}\sum_{k=1}^{N}(y(k) - r(k))^2 \tag{48}$$

and

$$MAE_{e_c} = \frac{1}{N}\sum_{k=1}^{N}(y(k) - r(k)) \tag{49}$$

where $y(k)$ is the time-varying system output, $r(k)$ is the desired value and the used number of observations N is 100.

In this simulation, the desired value, $r(k)$, is given in the following

$$r(k) = sin\,(2\pi k/25); \tag{50}$$

6.1 Simulation system using classical NN controller

In this section, we examine the effectiveness of the used classical neural network controller in the adaptive indirect control system. Indeed, in offline phase, using a reduced number of observations ($M = 3$) to find, either, the parameter initialization of the neural network parameters (w_{1j}, w_{ji}, v_{1j}, v_{ji}).

In online phase, at instant $(k+1)$, we use the input vector of the neural network controller $x_1 = [x_r(k), x_r(k-1), x_r(k-2), x_r(k-3), x_r(k-4)]^T$. The results of simulation are given by **Figures 3–5**.

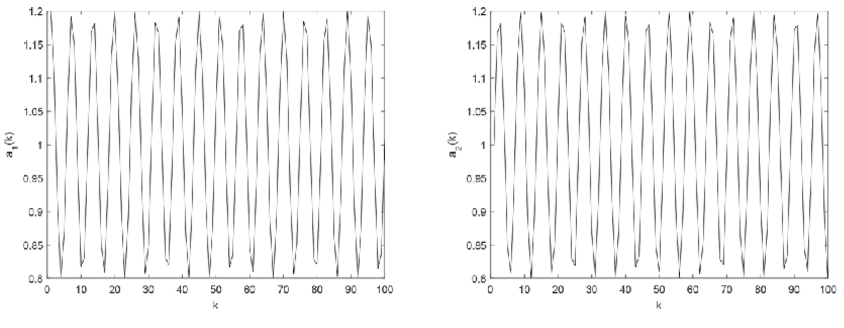


Figure 2.
$a_1(k)$ and $a_2(k)$ trajectories.

In this case, both neural network model and neural network controller consist of single input, 1 hidden layer with 8 nodes, and a single output node, identically. The used scaling coefficient is $\lambda = \lambda_c = 1$ and $\varepsilon_1 = \varepsilon_2 = 10^{-2}$.

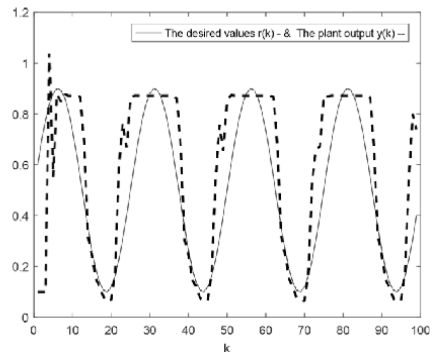


Figure 3.
The NN control system output and the desired values.

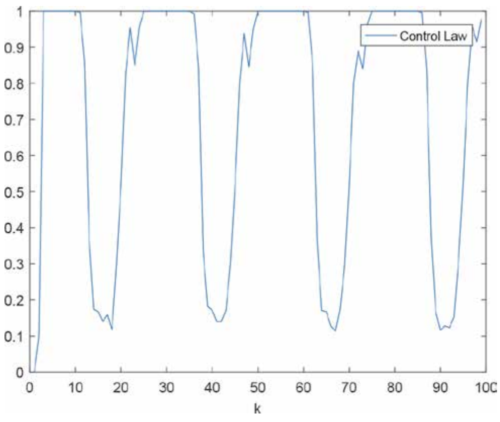


Figure 4.
The control law.

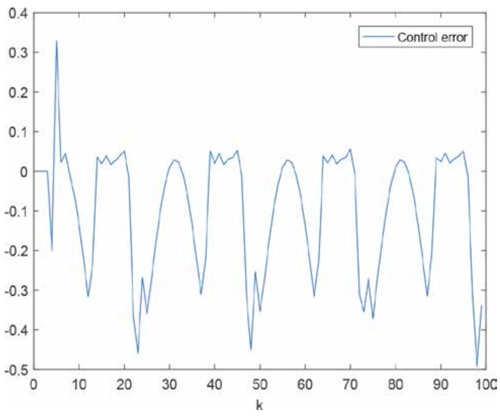


Figure 5.
The control error.

Using a multilayered perceptron architecture, three layers: one input layer, one hidden layer and one output layer. The result of simulation are given by the following figures.

6.2 Simulation system using PSO NN controller

The PSO parameters values are respectively the number of variables (m = 50), the population size (pop = 10), the maximum of inertia weight 0.9, the minimum of the inertia weight 0.4, the first acceleration factor (c1 = 2) and the second acceleration factor (c2 = 2).

Using a multilayered perceptron architecture, three layers: one input layer, one hidden layer and one output layer. The result of simulation are given by the following figures.

Figure 6 presents the control system output and the desired values. In this case, the neural network parameters of controller are optimized by PSO technique. A concordance between the desired values and the control system output is noticed, although the time-varying parameters.

However, **Figures** 7 and **8** present respectively the control law and the control error. These figures reveal that the PSO NN controller has smaller errors than the other controller.

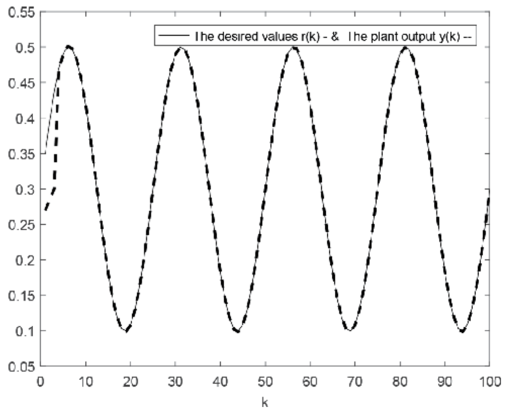


Figure 6.
The PSO NN control system output and the desired values.

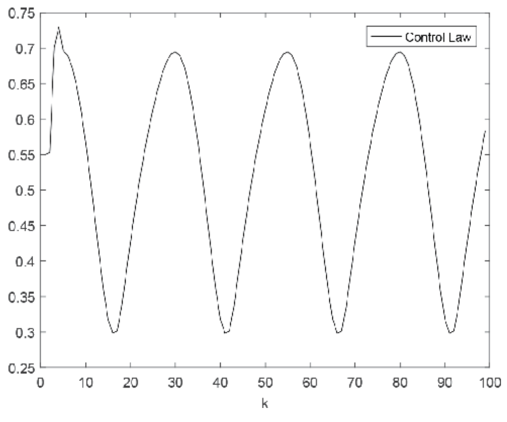


Figure 7.
The control law.

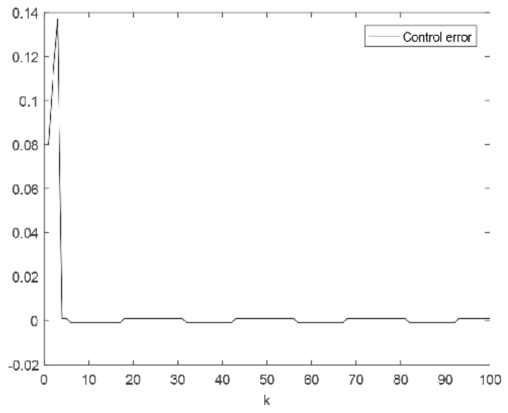


Figure 8.
The control error.

	NN controller	PSO NN controller
η	variable	variable
MSE_{e_c}	0.0349	$3.7730e-04$
$max\,(e_c)$	0.3291	0.1372
$min\,(e_c)$	-0.4907	$-9.9998e-04$
MAE_{e_c}	0.1305	0.0043
time (s)	323.829	100.926

Table 1.
The influence of the PSO technique in the control error.

Table 1 presents the influence of the PSO technique in the control error.

From **Table 1** we observe that, using the neural network controller, the PSO neural network controller has the smallest performance criteria in the control error $e_c(k)$. These results are shown in **Figures 6–8**.

6.3 Effect of disturbances

An added noise $v(k)$ is injected to the output of the time-varying nonlinear system in order to test the effectiveness of the proposed optimization technique of the neural network controller. To measure the correspondence between the system output and the desired value, a Signal Noise Ratio (SNR) is taken from the following equation:

$$SNR = \frac{\sum_{k=0}^{N} (y(k) - \bar{y})}{\sum_{k=0}^{N} (v(k) - \bar{v})} \tag{51}$$

with $v(k)$ is a noise of the measurement of symmetric terminal δ, $v(k) \in [-\delta, \delta]$, $\bar{y}$ and $\bar{v}$ are an output average value and a noise average value respectively. In this paper, the taken SNR is 5%.

Using the desired value $r(k)$, the sensitivity of the proposed neural network controller is examined in **Table 2**.

	NN controller	NN PSO controller
η_c	variable	variable
MSE_{e_c}	0.0351	$3.7728e-04$
MAE_{e_c}	00.1247	0.0042
$max\ (e_c)$	0.3123	0.1372
$min\ (e_c)$	0.5000	$-9.9999e-04$
time (s)	44.456972	337.728385

Table 2.
The influence of the PSO optimization in the control error.

From this table, we observe that, using the PSO as a method to optimize the parameters of neural network controller, we have got the smallest performance criteria in the control error.

According to the obtained simulation results, the lowest MSE_{e_c}, MAE_{e_c} and $max\ (e_c)$ are obtained using a combination between the neural network controller and the PSO technique, although the added disturbance in the system output and the time-varying parameters thanks to the PSO technique.

7. Conclusion

In this chapter, a comparative study between the neural network controller and neural network PSO controller is proposed and is applied with success in indirect adaptive control. For instance, the lowest MSE_{e_c}, MAE_{e_c}, $min\ (e_c)$ and $max\ (e_c)$ are obtained and it is proved that the PSO method is the best. The effectiveness of the proposed algorithm is successfully applied to single-input single-output system, with and without disturbances, and it proved its robustness to reject disturbances and to accelerate the speed of the learning phase of the neural model and neural controller.

Author details

Sabrine Slama, Ayachi Errachdi* and Mohamed Benrejeb
Tunis El Manar University, National Engineering School of Tunis, Department of Electrical Engineering, Le Belvédère B.P. 37, Tunis, Tunisia

*Address all correspondence to: errachdi_ayachi@yahoo.fr

References

[1] Slama S., Errachdi A. and Benrejeb M., Adaptive PID controller based on neural networks for MIMO nonlinear systems, Journal of Theoretical and Applied Information Technology, **97**, no. 2, pp. 361–371, 2019.

[2] Errachdi A. and Benrejeb M., Performance comparison of neural network training approaches in indirect adaptive control, International Journal of Control, Automation and Systems, **16**, no. 3, pp. 1448–1458, 2018.

[3] Saurabh G., Karali P. and Surjya K.P., Particle swarm optimization of a neural network model in a machining process, Sadhana **39**, Part 3, June 2014, pp. 533–548. Indian Academy of Sciences

[4] Zhou J., Duan Z., Li Y., Deng J. and Yu D., PSO-based neural network optimization and its utilization in a boring machine. J. Material Process Technol. **178**, pp. 19-–23, 2006.

[5] Elbeltagi E., Hegazy T. and Grierson D., Comparison among five evolutionary-based optimization algorithms. Advanced Eng. Informatics **19**, pp. 43-–53, 2005.

[6] Feng H.M., Self-generation RBFNs using evolutional PSO learning. Neurocomputing **70**, pp. 241–251, 2006.

[7] Karpat Y. and Ozel T., Hard turning optimization using neural network modelling and swarm intelligence. Transactions of NAMRI/SME **33**, pp. 179-–186, 2005.

[8] Zhang, R., Tao, J., Lu, R. and Jin, Q. Decoupled ARX and RBF neural network modeling using PCA and GA optimization for nonlinear distributed parameter systems. IEEE Transactions on Neural Networks and Learning Systems, **29**, no. 2, pp. 457–469, 2018.

[9] Stacey A., Jancic M. and Grundy I., Particle swarm optimization with mutation. Proceedings of IEEE, pp. 1425-1430, 2003.

[10] Zhao F., Ren Z., Yu D. and Yang Y., Application of an improved particle swarm optimization algorithm for neural network training. Proceedings of IEEE International Conference on Neural Networks and Brain, Beijing, China, pp. 1693-–1698, 2005.

[11] Asokan P., Baskar N., Babu K., Prabhaharan G. and Saravanan R., Optimization of surface grinding operations using particle swarm optimization technique. J. Manufacturing Sci. Eng. **127**, pp. 885–892, 2005.

[12] Haq A.N., Sivakumar K., Saravanan R. and Karthikeyan K., Particle swarm optimization (PSO) algorithm for optimal machining allocation of clutch assembly. Int. J. Advance Manufacturing Technol. **27**, pp. 865–869, 2006.

[13] Gaitonde V.N. and Karnik S.R., Minimizing burr size in drilling using artificial neural network (ANN)-particle swarm optimization (PSO) approach. J. Intelligent Manufacturing **23**, pp. 1783–1793, 2012.

[14] Navalertporn T. and Afzulpurkar N.V., Optimization of tile manufacturing process using particle swarm optimization. Swarm and Evolutionary Computation **1**, pp. 97–109, 2011.

[15] Samanta B. and Nataraj C., Use of particle swarm optimization for machinery fault detection. Eng. Appl. Artificial Intelligence, **22**, pp. 308–316, 2009.

[16] Malviya R. and Pratihar D.K., Tuning of neural networks using particle swarm optimization to model

MIG welding process. Swarm and Evolutionary Computation **1**, pp. 223-235, 2011.

[17] Garro B.A. and Vazquez R.A., Designing Neural Networks Using Particle Swarm Optimization, Research Article, Computational Intelligence in Neuroscience, Vol. 2015.

[18] Selvakumaran S., Parthasarathy S., Karthigaivel R. and Rajasekaran V., Optimal decentralized load frequency control in a parallel ac-dc interconnected power system through hvdc link using pso algorithm, Energy Procedia **14**, pp. 1849–1854, 2012.

[19] Shaher M., Reyad E. and Iqbal M.B. Tuning PID and $PI^{\lambda}D^{\delta}$ controllers using particle swarm optimization algorithm via El-Khazali's Approach, Proceedings of the 45th International Conference on Application of Mathematics in Engineering and Economics (AMEE'19) AIP Conf. Proc. 2172, 050003–1–050003-8; https://doi.org/10.1063/1.5133522 Published by AIP Publishing. 978–0–7354-1919–3/30.00

[20] Stimac G., Braut S. and Ziguli R, Comparative analysis of PSO algorithms for PID controller tuning, Chinese Journal of Mechanical Engineering, **27**, No. 5, 2014.

[21] Narendra K.S. and Parthasarthy K., Identification and control of dynamical systems using neural networks, IEEE Trans. on Neural Networks, **1**, 1, 4–27, 1990.

Chapter 2

Speech Enhancement Based on LWT and Artificial Neural Network and Using MMSE Estimate of Spectral Amplitude

Mourad Talbi, Riadh Baazaoui and Med Salim Bouhlel

Abstract

In this chapter, we will detail a new speech enhancement technique based on Lifting Wavelet Transform (*LWT*) and Artifitial Neural Network (*ANN*). This technique also uses the *MMSE* Estimate of Spectral Amplitude. It consists at the first step in applying the *LWT* to the noisy speech signal in order to obtain two noisy details coefficients, cD_1 and cD_2 and one approximation coefficient, cA_2. After that, cD_1 and cD_2 are denoised by soft thresholding and for their thresholding, we need to use suitable thresholds, $thr_j, 1 \leq j \leq 2$. Those thresholds, $thr_j, 1 \leq j \leq 2$, are determined by using an Artificial Neural Network (*ANN*). The soft thresholding of those coefficients, cD_1 and cD_2, is performed in order to obtain two denoised coefficients, cDd_1 and cDd_2 . Then the denoising technique based on *MMSE* Estimate of Spectral Amplitude is applied to the noisy approximation cA_2 in order to obtain a denoised coefficient, cAd_2. Finally, the enhanced speech signal is obtained from the application of the inverse of *LWT*, LWT^{-1} to cDd_1, cDd_2 and cAd_2. The performance of the proposed speech enhancement technique is justified by the computations of the Signal to Noise Ratio (*SNR*), Segmental *SNR* (*SSNR*) and Perceptual Evaluation of Speech Quality (*PESQ*).

Keywords: ANN, Lifting Wavelet Transform, Ideal thresholds, Soft thresholding, MMSE Estimate

1. Introduction

Numerous speech enhancement techniques have been developed in the previous years as speech enhancement is a core target in numerous challenging domains such as speech and speaker recognitions, telecommunications, teleconferencing and hand-free telephony [1]. In such applications, the goal is to recover a speech signal from observations degraded by diverse noises components [2]. The unusual noise components can be of various classes that are frequently present in the environment [3]. Many algorithms and approaches were proposed for resolving the problem of degraded speech signals [4–6]. Furthermore, methods of single or multi-microphones are proposed in order to ameliorate the behaviour of the speech enhancement approaches and also to reduce the acoustic noise components even in very noisy conditions [2]. The most well-known single channel approaches that are extensively known in speech enhancement application is the spectrum subtraction (SS) that requires only one channel signal [7]. It has been embedded in some high-quality mobile phones for the same application. Though, the SS approach is just appropriate for stationary noise environments [2]. Furthermore, it surely introduces music noise problem. In fact, the higher the noise is reduced, the greater the alteration is brought to the speech signal and accordingly the poorer the intelligibility of the enhanced speech is obtained [8, 9]. As a result, ideal enhancement can hardly be attained when the Signal to Noise Ratio (***SNR***) of the noisy speech is relatively low; below **5 *dB***. However, it has quite good result when the noisy speech ***SNR*** is relatively high; above **15 *dB*** [2]. The SS and other speech enhancement methods that are based on SS principal have ameliorated the decision directed (***DD***) methods in reducing the musical noise components [10–13]. Numerous algorithms that ameliorate the ***DD*** methods were suggested in [14]. In [15], a speech enhancement technique based on high order cumulant parameter estimation was proposed. In [16, 17], a subspace technique, which is based on Singular Value Decomposition (***SVD***) approaches was proposed; the signal is enhanced when the noise subspace is eliminated, and accordingly, the clean speech signal is estimated from the subspace of the noisy speech [2]. Another technique which was widely studied in speech enhancement application, is the adaptive noise cancellation (ANC) approach that was firstly suggested in [18, 19]. Moreover, most important speech enhancement methods employed adaptive approaches for getting the tracking ability of non-stationary noise properties [20, 21]. Numerous adaptive techniques were proposed for speech enhancement application, we can find time domain algorithm, frequency domain adaptive algorithms [22–26] or adaptive spatial filtering methods [27, 28] that frequently employ adaptive SVD methods in order to separate the speech signal space from the noisy one. Another direction of research combines the Blind Source Separation (BSS) methods with adaptive filtering algorithms for enhancing the speech signal and to cancel effectively the acoustic echo components [29–32]. This approach employs at least two microphones configuration in order to update the adaptive filtering algorithms. Also, a multi-microphone speech enhancement technique was proposed for the same aim and ameliorated the existing one-channel and two-channel speech enhancement and noise reduction adaptive algorithms [33, 34]. Numerous research works highlighted the problem of speech enhancement on a simple problem of mixing and unmixing signals with convolutive and instantaneous noisy observations [35–37]. In the last decade, a novel research direction has proven the efficacy of the wavelet domain as a novel effective mean that can ameliorates the speech enhancement approaches, and numerous algorithms and methods have been proposed for the same aim [38, 39]. In this chapter, we propose a novel speech enhancement technique based on Lifting Wavelet Transform (***LWT***) and Artifitial Neural Network (***ANN***) and also uses ***MMSE*** Estimate of Spectral Amplitude [40]. The presentation of this chapter is given as follows: after the introduction, we will deal with Lifting Wavelet Transform (***LWT***), in section 2. Section 3 is reserved to describe the proposed speech enhancement technique. Section 4 presents the obtained results and finally we conclude our work in section 5.

2. Lifting wavelet transform (*LWT*)

The Lifting Wavelet Transform (*LWT*) becomes a powerful tool for signal analysis due to its effective and faster implementation compared to the Discrete Wavelet Transform (*DWT*). In the domains of the signal denoising, signal compression and watermarking, the *LWT* permits to obtain better results compared to the *DWT* . The *LWT* permits to saves times and has a better frequency localization feature that overcomes the shortcomings of DWT. The Signal decomposition using LWT needs three steps: splitting, prediction and update.

3. The proposed technique

In this chapter, we propose a new speech enhancement technique based on Lifting Wavelet Transform (LWT) and Artifitial Neural Network (ANN). This technique also uses the $MMSE$ Estimate of Spectral Amplitude. It consists at the first step in applying the LWT to the noisy speech signal in order to obtain two noisy details coefficients, cD_1 and cD_2 and one approximation coefficient, cA_2. After that, cD_1 and cD_2 are denoised by soft thresholding and for their thresholding, we need to use suitable thresholds, $thr_j, 1 \leq j \leq 2$. Those thresholds, $thr_j, 1 \leq j \leq 2$, are determined by using an Artificial Neural Network (ANN). The soft thresholding of those coefficients, cD_1 and cD_2, is performed in order to obtain two denoised coefficients, cDd_1 and cDd_2. Then the denoising technique based on MMSE Estimate of Spectral Amplitude [40] is applied to the noisy approximation cA_2 in order to obtain a denoised coefficient, cAd_2. Finally, the enhanced speech signal is obtained from the application of the inverse of LWT, LWT^{-1} to cDd_1, cDd_2 and cAd_2. For each coefficient, $cD_j, 1 \leq j \leq 2$, the corresponding ideal $thr_j, 1 \leq j \leq 2$ is computed using the following MATLAB function:

```
function [thr] = Compute_Threshold (cc,cb).
R = [];
for(i = 1:length(cb)).
   r = 0;
   for(j = 1:length(cc)).
      r = r + (wthresh(cb(j),'s',abs(cb(i)))-cc(j)).^2;
   end;
      R = [R r];
end;
[guess,ibest] = min(R);
thr = abs(cb(ibest));
```

The inputs of this function are the clean details coefficient, cc and the corresponding noisy coefficient, cb. The output of this function is the ideal threshold, thr. Note that the couple (cc, cb) can be (ccD_1, cD_1) or (ccD_2, cD_2) where ccD_j *and* $cD_j, 1 \leq j \leq 2$ are respectively the clean details coefficient and the corresponding noisy details one at the level j.

In this chapter and as previously mentioned, the ideal threshold at level j is thr_j and is used for soft thresholding of the noisy details coefficient cD_j at that level. In this work, the used ANN is trained by a set of couples (P, T) where P is the input of this neural network and is chosen to be cD_j and T is the Target and is chosen to be the corresponding ideal threshold, thr_j at level j. Consequently, for computing a suitable threshold to be used in soft thresholing of $cD_j, 1 \leq j \leq 2$, we use one ANN so we have two $ANNs$ and two different suitable thresholds. Each of those $ANNs$ is constituted of two layers where the first one is a hidden layer and contains ten neurons having tansigmoid activation function and the second layer is the output one and contains one neurone having purelin activation function (**Figure 1**).

As shown in **Figure 1**, the input of this ANN is the noisy details coefficient at level $1 \leq j \leq 2$, $P = cD_j, 1 \leq j \leq 2$ and the desired output or Target, $T = thr_j, 1 \leq j \leq 2$. The activation functions tansigmoid and pureline are respectively expressed by Eqs. 1 and 2.

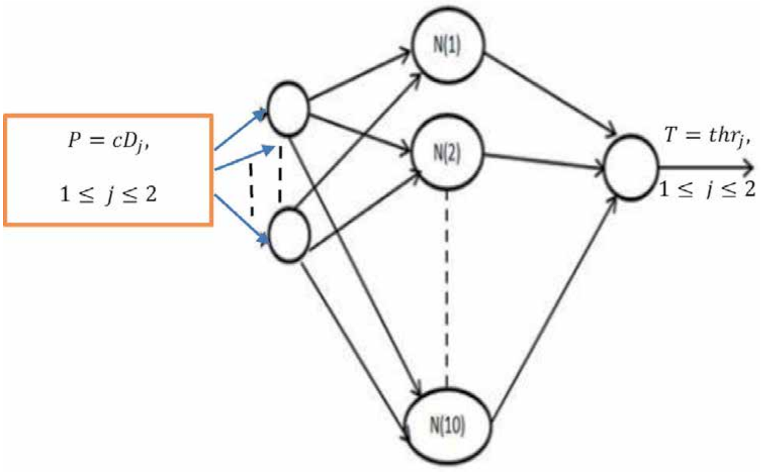


Figure 1.
The architecture of the ANN used in this work.

$$tansig\ (n) = 1/(1 + exp\ (-n)) \tag{1}$$

$$Purelin\ (n) = n \tag{2}$$

Generally, neural networks consist of a minimum of two layers (one hidden layer and another output layer). The input information is connected to the hidden layers through weighted connections where the output data is calculated. The number of hidden layers and the number of neurons in each layer controls the performance of the network. According to [41], there are no guidelines for deciding a manner for selecting the number of neurons along with number of hidden layers for a given problem to give the best performance. And it is still a trial and error design method [41].

For training each *ANN* used in this work we have employed 50 speech signals and 10 others used for testing those networks. Therefore, for training each used ANN, we used 50 couples of Input and Target (P, T). Evidently, the noisy speech signals used for the *ANNs* testing do not belong to the training database. The different parameters used for the training of the used ANNs are the epochs number which is equal to 5000, the momentum, μ or Mu which is equal to 0.1, the gradient minimum which is equal to $1e - 7$. The employed training algorithm is Leverberg-Marquardt.

In summary, the novelty of the proposed technique consists in applying the denoising technique based on MMSE Estimate of Spectral Amplitude [40]. Also, we apply the ANN for computing ideals thresholds to be used for thresholding of the noisy details coefficients obtained from the application of the *LWT* to the noisy speech signal.

4. Results and discussions

For the evaluation of the proposed technique, we have applied it to twenty Arabic speech signals pronounced by a male and female speakers. Those signals are corrupted in artificial manner with additive manner by two types of noise (White Gaussian and Car noises) at different values of *SNRi* before denoising.. The used Arabic speech signals (**Table 1**) are material phonetically balanced and they are sampled at 16 *kHz*.

Also for the evaluation of the proposed speech enhancement approach, we have applied the denoising technique based on *MMSE* Estimate of Spectral Amplitude [40]. This evaluation is performed in terms of Signal to Noise Ratio (*SNR*), the Segmental *SNR* (*SSNR*) and the Perceptual Evaluation of Speech Quality (*PESQ*). In **Tables 2**–**7** are listed the results obtained from the computations of SNRf (after

Female speaker	**Male speaker**
أحفظ من الأرض: **Signal1**	لا لن يذيع الخبر: **Signal 1**
أين المسا فرين: **Signal 2**	أكمل بالإسلام رسالتك: **Signal 2**
لا لم يستمتع بثمرها: **Signal 3**	سقطت إبرة: **Signal 3**
سيؤذيهم زماننا: **Signal 4**	من لم ينتفع: **Signal 4**
كنت قدوة لهم: **Signal 5**	غفل عن ضحكاتها: **Signal 5**
ازار صائما: **Signal 6**	و لماذا نشف مالهم: **Signal 6**
كال و غبط الكبش: **Signal 7**	أين زوايانا و قانوننا: **Signal 7**
هل لذعته بقول:**Signal 8**	صاد الموروث مدلعا: **Signal 8**
عرف واليا و قائدا: **Signal 9**	نبه آبائكم: **Signal 9**
خال بالنا منكما: **Signal 10**	أظهره و قم: **Signal 10**

Table 1.
The list of the employed Arabic speech sentences.

SNRi (dB)	**SNRf (dB)**	
	The Denoising approach	
	The proposed speech enhancement technique	**The denoising technique based on MMSE Estimate of Spectral Amplitude [40]**
−5	**8.3650**	**7.1431**
0	**13.0857**	**11.6110**
5	**16.9010**	**15.5721**
10	**19.8933**	**18.8719**
15	**22.3135**	**21.7972**

Table 2.
Results in term of SNR (signal 4 (female voice) corrupted by Gaussian white noise).

SNRi (dB)	**SSNR (dB)**	
	The Denoising approach	
	The proposed speech enhancement technique	**The denoising technique based on MMSE Estimate of Spectral Amplitude [40]**
−5	**−0.0954**	−0.7089
0	**2.5997**	1.7725
5	**4.7373**	3.9719
10	**6.8038**	5.9329
15	**9.5324**	8.7158

Table 3.
Results in term of SSNR (signal 4 (female voice) corrupted by Gaussian white noise).

denosing), of SSNR and PESQ and this for the proposed technique and The denoising technique based on MMSE Estimate of Spectral Amplitude [40].

According to those results listed in **Tables 2–7**, the best results are in Red colour. In terms of *SNRf* (After denoising), and *SSNR*, the best results are those obtained

SNRi (dB)	**PESQ**	
	The Denoising approach	
	The proposed speech enhancement technique	**The denoising technique based on MMSE Estimate of Spectral Amplitude [40]**
−5	**1.3225**	**1.3755**
0	**1.5935**	**1.6320**
5	**1.8812**	**1.8977**
10	**2.2016**	**2.2311**
15	**2.5147**	**2.6079**

Table 4.
Results in term of PESQ (signal 4 (female voice) corrupted by Gaussian white noise).

SNRi (dB)	**SNRf (dB)**	
	The Denoising approach	
	The proposed speech enhancement technique	**The denoising technique based on MMSE Estimate of Spectral Amplitude [40]**
−5	**5.8737**	4.2192
0	**9.8414**	8.3451
5	**14.1647**	12.6024
10	**18.5308**	17.4120
15	**22.5102**	21.4578

Table 5.
Results in term of SNR (signal 2 (male voice) corrupted by car noise).

SNRi (dB)	**SSNR (dB)**	
	The Denoising approach	
	The proposed speech enhancement technique	**The denoising technique based on MMSE Estimate of Spectral Amplitude [40]**
−5	**0.2145**	−1.1347
0	**2.7478**	1.7861
5	**5.6644**	4.7166
10	**8.8942**	7.8228
15	**11.9663**	10.9850

Table 6.
Results in term of SSNR (signal 8 (male voice) corrupted by car noise).

from the application of the proposed speech enhancement technique. However, in term of PESQ, the denoising technique based on *MMSE* Estimate of Spectral Amplitude [40] is slightly better than the proposed technique.

In **Figures 2–5** are illustrated some examples of speech enhancement using the proposed technique.

These Figures show the efficiency of the proposed speech enhancement technique. In fact, it permits to reduce considerably the noise while conserving the original signal and this especially when the *SNRi* is higher (5, 10 and 15 dB).

In our future work and in order to improve this proposed speech enhancement technique, we will use a Deep Neural Network instead (DNN) instead of a simple ANN and other transforms such as Empirical Mode Decomposition (EMD).

SNRi (dB)	PESQ	
	The Denoising approach	
	The proposed speech enhancement technique	The denoising technique based on MMSE Estimate of Spectral Amplitude [40]
−5	2.2837	2.4021
0	2.5999	2.7163
5	2.8709	3.0184
10	3.1190	3.2461
15	3.3590	3.4789

Table 7.
Results in term of PESQ (signal 8 (male voice) corrupted by car noise).

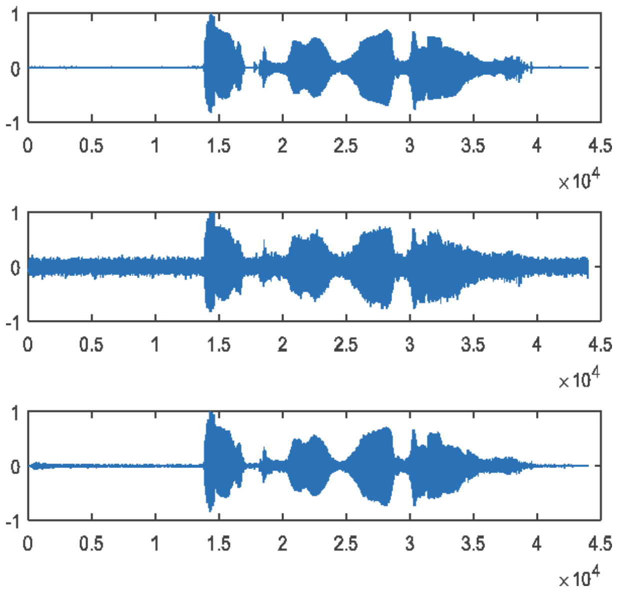


Figure 2.
*An example of speech enhancement applying the proposed technique: Signal 4 (pronounced by a female voice (**Table 1**) corrupted by Gaussian white noise with SNRi = 10dB (before enhancement)). After enhancement we have: SNRf = 19.8933, SSNR = 6.8038 and PESQ = 2.2016.*

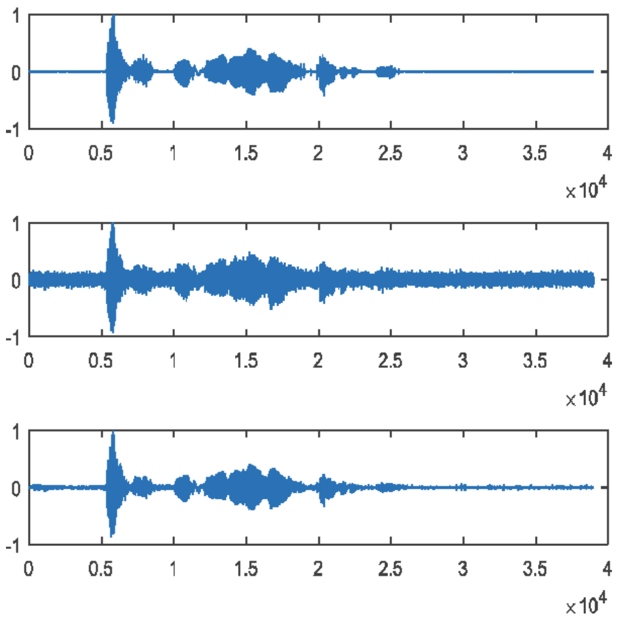


Figure 3.
*An example of speech enhancement applying the proposed technique: Signal 1 (pronounced by a male voice (**Table 1**) corrupted by Gaussian white noise with SNRi = 5dB (before enhancement)). After enhancement we have: SNRf = 13.7710 , SSNR = 0.7135 and PESQ = 2.2350.*

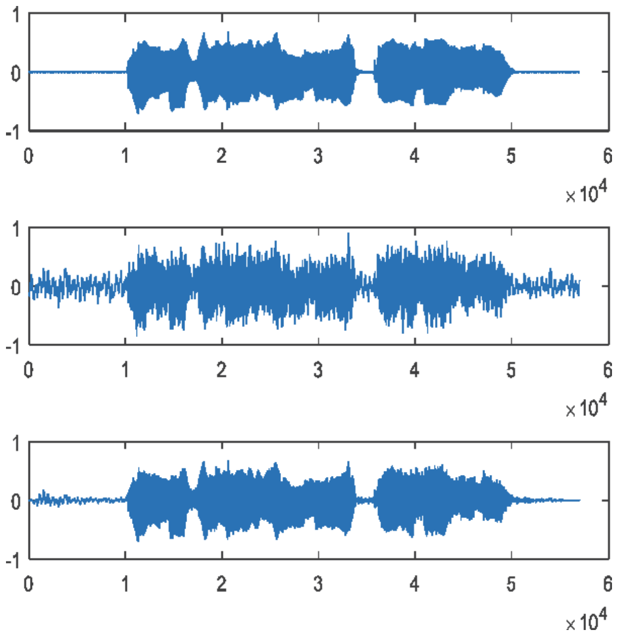


Figure 4.
*An example of speech enhancement applying the proposed technique: Signal 7 (pronounced by a male voice (**Table 1**) corrupted by car noise with SNRi = 5dB (before enhancement)). After enhancement we have: SNRf = 15.1244, SSNR = 8.7594 and PESQ = 3.3304.*

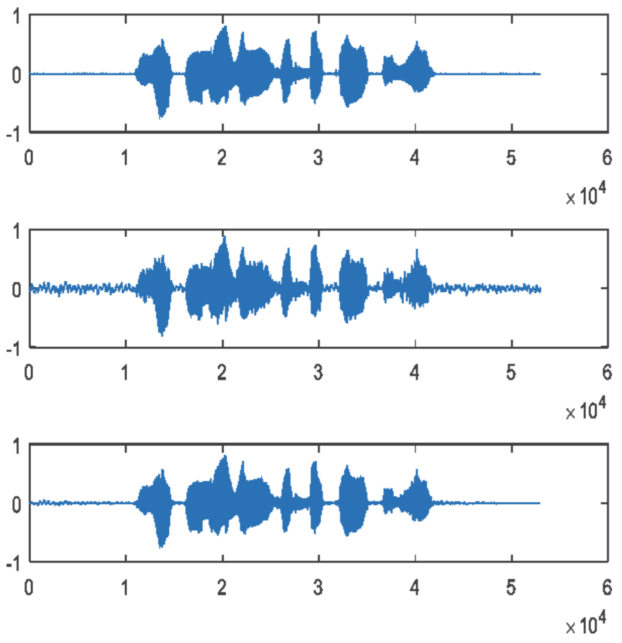


Figure 5.
*An example of speech enhancement applying the proposed technique: Signal 5 (pronounced by a male voice (**Table 1**) corrupted by car noise with SNRi = 10dB (before enhancement)). After enhancement we have: SNRf = 18.8848, SSNR = 6.4497 and PESQ = 3.5469.*

5. Conclusion

In this chapter, we will detail a new speech enhancement technique based on Lifting Wavelet Transform (LWT) and Artifitial Neural Network (ANN). This technique also uses the $MMSE$ Estimate of Spectral Amplitude. It consists at the first step in applying the LWT to the noisy speech signal in order to obtain two noisy details coefficients, cD_1 and cD_2 and one approximation coefficient, cA_2. After that, cD_1 and cD_2 are denoised by soft thresholding and for their thresholding, we need to use suitable thresholds, $thr_j, 1 \leq j \leq 2$. Those thresholds, $thr_j, 1 \leq j \leq 2$, are determined by using an Artificial Neural Network (ANN). The soft thresholding of those coefficients, cD_1 and cD_2, is performed in order to obtain two denoised coefficients, cDd_1 and cDd_2. Then the denoising technique based on $MMSE$ Estimate of Spectral Amplitude is applied to the noisy approximation cA_2 in order to obtain a denoised coefficient, cAd_2. Finally, the enhanced speech signal is obtained from the application of the inverse of LWT, LWT^{-1} to cDd_1, cDd_2 and cAd_2. The performance of the proposed speech enhancement technique is justified by the computations of the Signal to Noise Ratio (SNR), Segmental SNR ($SSNR$) and Perceptual Evaluation of Speech Quality ($PESQ$).

Author details

Mourad Talbi[1]*, Riadh Baazaoui[2] and Med Salim Bouhlel[3]

1 Laboratory of Nanomaterials and Systems for Renewables Energies (LaNSER), Center of Researches and Technologies of Energy of Borj Cedria, Tunis, Tunisia

2 Faculty of Sciences of Tunis, University of Tunis El-Manar, Tunisia

3 Sciences and Technologies of Images and Telecommunications, Higher Institute of Electronics and Telecommunication of Sfax, Sfax University Sfax, Tunisia

*Address all correspondence to: talbi1969@yahoo.fr

References

[1] Loizou, P. C. (2007). Speech enhancement: Theory and practice (pp. 589–599). Boca Raton, FL: Taylor and Francis.

[2] Mohamed, D. (2018). An efficient wavelet-based adaptive filtering algorithm for automatic blind speech enhancement. *International Journal of Speech Technology*, 21:355–367.

[3] Djendi, M., Scalart, P., & Gilloire, A. (2013). Analysis of two-sensor forward BSS structure with post-filters in the presence of coherent and incoherent noise. Speech Communication, 55(10), 975–987.

[4] Dixit, S., & Mulge, M. Y. (2014). Review on speech enhancement techniques, International Journal of Computer Science and Mobile Computing, 3(8), 285–290

[5] Bactor, P., & Garg, A. (2012). Different techniques for the enhancement of the intelligibility of a speech signal. International Journal of Engineering Research and Development, 2(2), 57–64.

[6] Scalart, P., Filho, J. (1996). Speech enhancement based on a priori signal to noise estimation. In International Conference on Acoustics, Speech, and Signal Processing. pp. 629–632.

[7] Boll, SF (1979). Suppression of acoustic noise in speech using spectral subtraction. IEEE Transactions on Acoustics, Speech and Signal Processing, 27, 113–120.

[8] Zhang, Y., & Zhao, Y. (2012). Real and imaginary modulation spectral subtraction for speech enhancement. *Journal on Speech Communication*, 55(6), 509–522.

[9] Cappé, O. (1994). Elimination of the musical noise phenomenon with the Ephraïm and Malah noise suppressor. *IEEE Transactions on Speech Audio Processing*, 2(2), 345–349.

[10] Ephraim, Y., & Malah, D. (1984). Speech enhancement using a minimum mean-square error short-time spectral amplitude estimator. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(6), 1109–1121.

[11] Ephraim, Y., & Malah, D. (1985). Speech enhancement using a minimum mean-square error log-spectral amplitude estimator. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP33(2), 443–445.

[12] Ephraim, Y., LevAri, H., Roberts, W. J. J. (2014). A brief survey of speech enhancement. *IEEE Signal Processing Letters*, 10, 104–106.

[13] Selva Nidhyananthan, S., Shantha Selva Kumari, R., & Arun Prakash, A. (2014). A review on speech enhancement algorithms and why to combine with environment classification. *International Journal of Modern Physics C*, 25(10), 210–225.

[14] Wolfe, P. J., & Godsill, S. J. (2003). Efficient alternatives to the Ephraim and Malah suppression rule for audio signal enhancement. *EURASIP Journal on Applied Signal Processing*, 10, 1043–1051.

[15] Dong, J., Wei, X. P., & Zhang, Q. (2009). Speech enhancement algorithm based on high-order Cumulant parameter estimation. *International Journal of Innovative Computing information and Control*, 5, 2725–2733.

[16] Davila, C. E. (1984). A subspace approach to estimation of autoregressive parameters from noisy measurements. IEEE Transaction on Signal processing, 46, 531–534.

[17] Doclo, S., & Moonen, M. (2002). GSVD-based optimal filtering for signal and multi-microphone speech enhancement. *IEEE Transaction on Signal processing*, 50, 2230–2244.

[18] Widrow, B., Goodlin, R. C. (1975). Adaptive noise cancelling: Principles and applications. *Proceedings of the IEEE*, 63, 1692–1716.

[19] Widrow, B., & Stearns, S. D. (1985). Adaptive signal processing, Upper Saddle River: Prentice-Hall.

[20] Lee, K. A., & Gan, W. S. (2004). Improving convergence of the NLMS algorithm using constrained subband updates. IEEE Signal Processing Letters, 11(9), 736–739.

[21] Weinstein, E., Feder, M., & Oppenheim, A. V. (1993). Multi-channel signal separation by decorrelation. IEEE Transactions on Speech Audio Processing, 1(4), 405–413.

[22] Plapous, C., Marro, C., Scalart, P., Mauuary, L., & Two-Step, A. (2004). Noise reduction technique. In IEEE International Conference on Acoustics, Speech, Signal Processing, Montral, Quebec Canada, 1, pp. 289–292.

[23] Al-Kindi, M. J., & Dunlop, J. (1989). Improved adaptive noise cancellation in the presence of signal leakage on the noise reference channel. Signal Processing, 17(3), 241–250.

[24] Plapous, C., Marro, C., Scalart, P. (2005). Speech enhancement using harmonic regeneration, In IEEE International Conference on Acoustics, Speech, Signal Processing, Philadelphia, PA, USA, 1, pp. 157–160.

[25] Djendi, M., Khemies, F., & Morsli, A. (2015). A Frequency Domain Adaptive Decorrelating Algorithm for Speech Enhancement. In International Conference on Speech and Computer, SPECOM 2015, pp. 51–54.

[26] Djendi, M., Bensafia, S., & Safi, M. (2016). A frequency co-channel adaptive algorithm for speech quality enhancement, In International Conference on Engineering and MIS (ICEMIS).

[27] Tong, R., Bao, G., & Ye, Z. (2015). A higher order subspace algorithm for multichannel speech enhancement. IEEE Signal Processing Letters, 22(11), 2004–2008.

[28] Goldsworthy, R. L. (2014). Two-microphone spatial filtering improves speech reception for cochlear-implant users in reverberant conditions with multiple noise sources. Trends in Hearing, 18, 1–13.

[29] Van Gerven, S., & Van Compernolle, D. (1992). Feed forward and feedback in symmetric adaptive noise canceller: Stability analysis in a simplified case. In European Signal Processing Conference, Brussels, Belgium. pp. 1081–1084.

[30] Djendi, M., Scalart, P., & Gilloire, A. (2006). Noise cancellation using two closely spaced microphones: Experimental study with a specific model and two adaptive algorithms. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 3, pp. 744–747.

[31] Zoulikha, M., & Djendi, M. (2016). A new regularized forward blind source separation algorithm for automatic speech quality enhancement. Applied Acoustics, 112, 192–200

[32] Jin, Y. G., Shin, J. W., & Kim, N. S. (2014). Spectro-temporal filtering for multichannel speech enhancement in short-time Fourier transform domain. IEEE Signal Processing Letters, 21(3), 352–355.

[33] Benesty, J., & Cohen, I. (2017). Multichannel speech enhancement in the STFT domain. In J. Benesty, & I. Cohen (Eds.), Canonical correlation analysis in speech enhancement, Springer briefs in electrical and computer engineering (pp. 79–101). New York: Springer.

[34] Lee, G., & Dae Na, S. (2016). Seong K2, Cho JH3, Nam Kim M4. Wavelet speech enhancement algorithm using exponential semisoft masks filtering. Bioengineered, 7(5), 352–356.

[35] Jutten, C., & Herrault, J. (1991). Blind separation of sources: an adaptive algorithm based on neuromimetic architecture. Signal Processing, 24, 1–10.

[36] Nguyen Thi, H. L., & Jutten, C. (1995). Blind sources separation for convolutive mixtures. Signal Processing, 45, 209–229.

[37] Mansour, A., Jutten, C., & Loubaton, P. (1996). Subspace method for blind separation of sources and for a convolutive mixture model. Signal processing VIII, theories and applications (pp. 2081–2084).

[38] Bouzid, A., & Ellouze, N. (2016). Speech enhancement based on wavelet packet of an improved principal component analysis. Journal Computer Speech and Language, 35, 58–72.

[39] Ghribi, K., Djendi, M., & Berkani, D. (2016). A wavelet-based forward BSS algorithm for acoustic noise reduction and speech enhancement. Applied Acoustics, 105, 55–66.

[40] Timo Gerkmann, and Richard C. Hendriks. Unbiased MMSE-Based Noise Power Estimation With Low Complexity and Low Tracking Delay. IEEE Transactions on Audio, Speech, And Language Processing, Vol. 20, No. 4, May 2012.

[41] http://www.aast.edu/en/openfiles/opencmsfiles/pdf_retreive_cms.php?disp_unit=351/Hatem%20Diab-Thesis.pdf

Chapter 3

The Digital Twin of an Organization by Utilizing Reinforcing Deep Learning

Marko Kesti

Abstract

Chapter deals with latest knowledge on deep reinforcement learning in the context of organizational management. Article presents reinforcement learning (RL) as a tool for the manager on the path to learning winning behavior in the complex environment of organization management. Organization management has wicked learning challenges because agents are under biases that prevent understanding the phenomenon of delayed reward. Therefore, the digital simulation with RL is effective forming breakthrough learning results. Human capital management theories provide architecture in creating organization digital twin where agent can practice management actions effect on business economics and staff wellbeing. Utilizing RL algorithms, it is possible to foster behavior for creating sustainable competitive advantage – this means the Nash equilibrium between profit and staff wellbeing. In this digital twin there is AI learning assistant as a teacher that provides demonstrations on how to act so that the delayed reward is good in the future. The article explains game theoretical approach that is the foundation for creating management deep learning AI system. Human agent at the organization is playing the game of Strategic Stochastic Bayesian Nonsymmetric Signaling game in co-operative or non-cooperative way and at zero-sum or general sum game mind-set.

Keywords: Reinforcement learning, Digital twin, QWL, Management, Game theory

1. Introduction

The state-of-the-art management literature focuses on the qualitative characteristics of management, bringing empirical evidence-based models for improving organization performance. However, the management models that appear in the literature do not consider the individual complexity of organizations, thus limiting the reproducibility of good results. The organization digital twin (ODT) used in the article demonstrates the potential of RL-AI to analyze and quantify complex phenomena related to organizational behavior. In this article we study model-driven reinforcement learning AI as a new method in improving organization performance at complex environment.

There are two main categories of artificial intelligence (AI): data-driven and model-driven. Data-driven AI uses data in finding correlations and forecasting the future. In model-driven AI there is model that simulates the environment. Reinforcement learning (RL) focus is in learning and finding behavior which gives best value and reward. When RL is utilized at model-driven AI the model simulates the behavior's effect in the value. The agent tries to learn the best behavior by following the model's reward signals. Thus, the behavior of the agent determines the result, not the data of the past.

Game Theory is a branch of mathematics that are used to model the strategic interaction between different players on a context of predefined environment. At management game theory there is predefined organization environment where the players are leaders and team members as workers or employees. Each player has incentives that drives their behavior in the game. Management game is non-symmetric because leader has specific and non-changeable characteristic compared to workers. Workers are motivated in maintaining and improving their work performance and personal self-esteem. Team leaders are motivated in maintaining and improving team performance, which is related to team leader personal profit incentives. Team leader knows that team performance is essential for achieving team profit targets. Workers know that their personal incentives will improve if their work performance is good. Thus, if there are problems at work the rational policy would be to tell the problems to supervisor so that problems can be solved. In addition, solving problems may improve workers self-esteem, having hidden psychological incentive. This organization environment form state space for strategic-Bayesian-stochastic-nonsymmetric-signaling game.

Nash equilibrium is a concept of game theory where optimal outcome is the balance where all players incentives are considered and fulfilled in optimal way. If team leader gives positive feedback for raising the possible problems, it will have positive effect on workers' self-esteem, fostering workers policy to inform the problems by signals. Solving the problems will improve group performance which foster leader's policy to encourage workers signaling game. This way workers and supervisor may find equilibrium of policies (strategies) which lead to general-sum game where optimal and sustainable team profit performance is achieved. However, this article explains why this optimal equilibrium is difficult to achieve in reality. Bersin [1] study reveal that 89% of managers think that leadership is important issue, but current leadership programs bring only minor value in improving leadership quality. This article argues that modern reinforcement learning artificial intelligence gives one solution in solving leadership challenge.

In addition to administrative role, the HR management has important function on adding competitive business value to an organization management (for example see references [2–6]). Managers need predictive measurements that indicate how business is developing and how to improve it. Human assets are essential for creating competitive advantages, thus interest in performance management has increased. Fleetwood and Hesketh [7] argue that researchers should better understand the complexity of the organization environment and seek to open a "black box" of causal relationships between human resources and organizational performance rather than offering simplified solutions.

Several studies indicate that employee psychological well-being has tendency to predict business value of an organization (for example see references [2, 8]). However, management can be confused of how to improve well-being and how much effort should be invested in well-being development at different situations to gain sufficient payback. Research reveals that organizations expect artificial Intelligence to help reducing managerial biases related to human issues and to improve productivity and employee experience [9]. Beside the hopes, researchers are also concerned that artificial intelligence may cause serious harm if the organization context is oversimplified by using data driven machine learning algorithms [10]. This article argues that AI can help solving difficult management problems related to human biases. One of the most promising new technology is Digital Twin that uses simulation model driven AI. "To build an efficacious Digital Twin, it's important to first agree what problem needs to be solved or what opportunity needs to be explored and how accurate do the predictions need to be" [11].

Human competencies, for example leadership and working skills, have certain causalities to long term productivity. It seems that human competence has three performance-driving characteristics that can be described according to motivation theory as feelings of safety, team culture, and passion for work. It is clear that a passion for work affects a person's performance in a very different way than for example occupational safety issues. In addition, human is a psychophysical entity tied to his own situation. Therefore, the combination of all motivational drivers determines performance [12].

First, we have to study human capital productivity, which includes working time and the utilization of intangible human assets. Human intangible assets refer to performance on how effectively is the working time utilized, and how much value a person produces at each working hour. An employee may work for eight hours a day, but out of that working time, how much is actually used effectively in creating value? This basic understanding of how each employee produces value needs to be recognized before any reliable simulation analytics can be made.

2. Concept of organization digital twin

At this article the organization digital twin (ODT) refers to the mathematical environment that simulate organization human capital productivity. To be able to simulate the reality the digital twin must meet following requirements:

- Markov property: The future is independent of the past given the current situation.

- The environment state can be verified from measuring the reality.

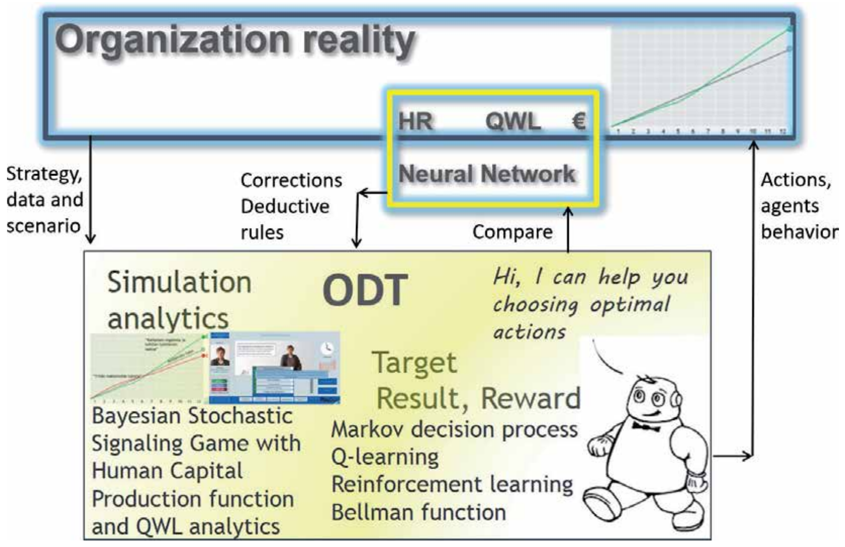

Organization reality
HR
QWL
€
Neural Network
Strategy, data and scenario
Corrections Deductive rules
Compare
Actions, agents behavior
Simulation analytics
ODT
Hi, I can help you choosing optimal actions
Target
Result, Reward
Bayesian Stochastic Signaling Game with Human Capital Production function and QWL analytics
Markov decision process
Q-learning
Reinforcement learning
Bellman function

Figure 1.
Concept of organization digital twin.

Markov property means that the future is not determined by the past data, thus supervised learning regression analytics cannot be solely applied in creating ODT. Markov rule is one backbone for creating ODT digital twin and for utilizing Reinforcement Learning where the behavior of the agents determines the future.

The state transition from state to state follows Markov chain where all necessary information is transferred from past to the present. Therefore, the probability of transition from the current state to the next state depends only on the current data and the activity of the players. In the digital twin, this current data must be able to determine the reality presented by the twin. The data in the twin can be measured and verified from reality, thus creating a feedback loop from the real world. This model verification against reality is also necessary for learning purposes so that ODT can learn to refine the transition functions to match the real world (**Figure 1**).

3. Opening the "black-box" of human capital productivity

New science provided theoretical framework for creating ODT for modeling organization human capital productivity. First, there should understand how employees produce economic value. The theory of Quality of Working Life (QWL) determines the effective working time-share from the time spend at work. According the human capital production function the staff effective working time multiplied by K-coefficient produces customer value that is measured by revenue. The coefficient K describes the business branch, tangible investments and business logic. QWL improvement requires HR-development that increase auxiliary working time, thus reducing time for work [12] (**Figure 2**).

The human capital production function can be written in function where revenue is the production volume according the Equation [13]:

$$\mathrm{R} = \mathrm{K} * \mathrm{L} * TWh * (1 - Ax) * QWL \tag{1}$$

and

$$\text{Profit} = \text{R} - \text{Variable costs} - \text{Staff costs} - \text{other costs} \tag{2}$$

Where
R = Revenue [$].
K = Coefficient for effective working time revenue relation, HR business ratio [$/h].
L = Labor capacity in full-time equivalent [pcs].
TWh = Theoretical yearly working time [h].
QWL = quality of working life, indicating human capital intangible asset utilization (0–100%).
Ax = The auxiliary working time of the total theoretical working time (vacation, absence, family leave, orientation, training, HR practices, and HRD) [%].
(1 – Ax) = (100% – Ax) = Time available for actual work (time spent at work)
(1 – Ax) * QWL = Effective working time from the theoretical working time.

It should be noted that other working time includes so-called internal error factors such as waiting, searching, correcting and unnecessary work. These are symptoms of different kinds of development needs that the team has noticed either hidden or conceptual.

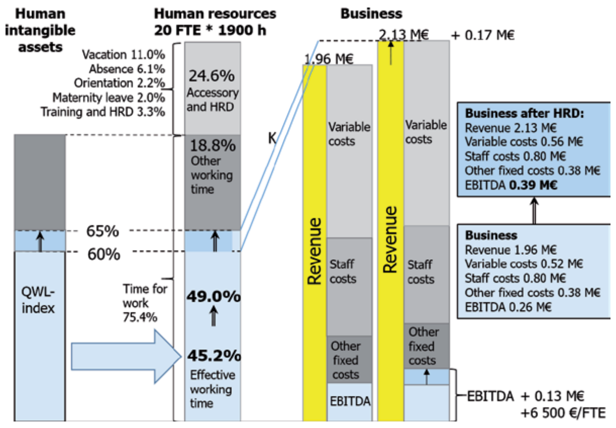


Figure 2.
Illustration of profit team human capital production function.

When manager does efficient team development there can be increase in effective working time. In addition, if absence and staff turnover is high the development may reduce those, thus increase time for work. This way effective team development will increase effective working time, and have good effect on profit. However, at short notice the development will increase auxiliary working time and reduce time for work, thus will reduce both revenue and profit. The development of human capital involves the phenomenon of investment, which requires some sacrifice in order to gain delayed rewards. When investment phenomenon is involved in the agents' actions there is possible to utilize Q-learning function.

Q-learning is a mathematical method for analyzing behavioral learning points in a simulation model that considers short- and long-term rewards. Nash equilibrium is the result where Q-learning settles to a certain level where the model environment is stable and no player can improve his pay-off [14]. In this case, equilibrium is achieved with a behavior in which both QWL and profit mature to a certain level. Both QWL and profit are management game agents' rewards, which in short-term may be contradictory because improving QWL reduces profit in short-term. This article shows that there are several states of equilibrium in a leadership game.

In most traditional well-being and commitment surveys, scores are averages of factors that are not individually relevant to the whole. Thus, the result is for example engagement index that does not necessarily tell what and how to improve and what impact the improvement would likely have. Traditional well-being surveys with average scores are oversimplified when measuring human performance. For ODT perspective, it is essential that the staff performance is determined realistically. It affects to the rewards and transition functions of agents' behaviors. Therefore, it is essential to describe the theory of QWL.

It seems evident that human performance is rather complex phenomenon, consisting several motivation theoretical aspects that cannot be included at simplified statistical staff survey analytics. Therefore, we have utilized motivation theories of Alderfer [15], Antonovsky [16], Kano [17] and Herzberg [18] in creating advanced human performance theory that meets the contribution of main scientists and forms practical QWL index for performance analytics. QWL index includes three self-esteem categories, which each has unique effect on performance.

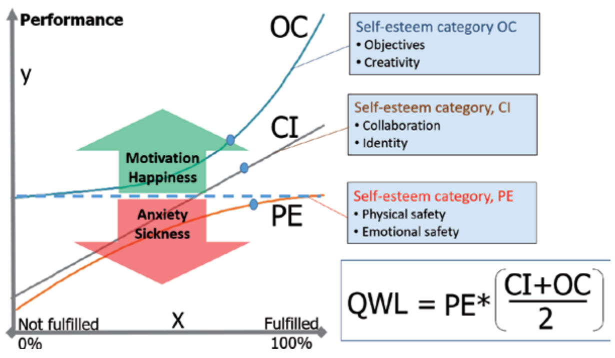


Figure 3.
The theory of QWL.

The self-esteem categories:

- Physical and emotional safety (PE);
- Collaboration and identity (CI); and
- Objectives and creativity (OC).

Chosen categories and their effect on performance form the theory of QWL index. It is also important to know that in addition that QWL index is production parameter, it has also logical connection to customer satisfaction (see [17]) (**Figure 3**).

Finally, the QWL index is the combination of all three self-esteem factors according the following equation:

$$QWL = PE(x_1) * \left[\frac{(CI(x_2) + OC(x_3))}{2} \right] \tag{3}$$

where

QWL is calculated using the quality of working life index (0 … 1).

PE(x_1) is the value of the function of physical and emotional safety.

CI(x_2) is the value of the function of collaboration and identity.

OC(x_3) is the value of the function of objectives and creativity.

The functions of the self-esteem categories are adjusted so that the final QWL result is always between 0 and 100% [12].

4. Defining management-game for ODT

Bayesian theorem with stochastic game is utilized in defining management-game for ODT. Using game theory, we can model the strategic interaction between different players (agents) in a predefined environment. Our management-game is multi-agent game for the profit unit where the agents are workers and manager. The concept is non-symmetric because manager (team leader) and workers have different roles and their reward characteristics differ. Workers are motivated to maintain and improve their self-esteem (QWL). In addition, there might be some hidden motivation drivers. Team leader motivation drivers are unit profit and possible personal incentives, which may be hidden (e.g. biases). In our game the focus is profit-unit manager's behavior and learning.

At Nash equilibrium the optimal outcome of the game is one where no agent wants to deviate from the chosen policy because that seems to be parallel with opponents' policy. Workplace problems have reducing tendencies on workers' self-esteem, thus decreasing QWL as a production parameter. Management practices have tendencies to improve QWL, but each action will reduce short-term profit. Manager's strategy hypothesis guides the actions at different state events. When the consequence data of action tendencies update the status after each Markov-sequence, the player can update the management strategy, which further controls the next actions. Bayesian probability is related to player subjective behavior, relying on the phenomenon that rational thinking will probably lead to optimal result as the new information comes available [19].

The manager should learn the optimal leadership strategy without knowing the exact reward function or state transition function. This approach is called stochastic model-free reinforcement learning and can be defined with the Nash Q-learning approach. The leader has prior-believe about the state of nature of profit-unit business situation and expected future reward. The uniqueness of the game comes from the fact that it has predictive features that allow for the use of reinforcing learning artificial intelligence for learning Nash equilibrium between staff QWL and organization profit.

Management game is signaling game since workers give essential signals about possible workplace problems that may threaten their self-esteem (QWL) and therefore team performance. Workers preference strategy is to give their leader signals about the problems. In simplified digital team leaders' learning-game the worker's strategy may be stationary, meaning that workers behavior may be chosen in advance when the events scenario is known.

Team leader, as an agent of the management game, is responsible for team profit performance that is the outcome of producing customer value measured by revenue. Agent registers workers' signals and makes own prior belief for the strategy. Agent monitors also scorecards from business outcomes of monthly and cumulative profit, and forms a prior believe policy on how to act to these measures. Agent is rewarded by the profit at each month and cumulative profit at the end of the year. After each state transition the agent will get profit signals and QWL signals from the worker's response from the state change at workers QWL. State-change signals and reward results may cause changes at the preference strategy of the agent for the next sequence (Markov sequence [19]) (**Figure 4**).

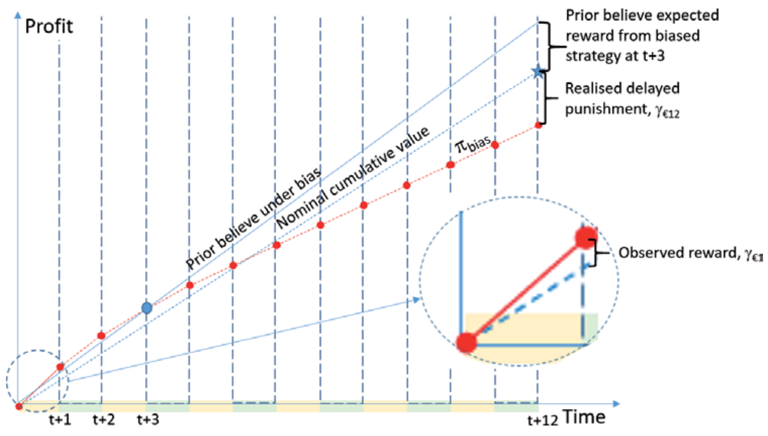

Profit
Prior believe expected reward from biased strategy at t+3
Realised delayed punishment, $\gamma_{\epsilon 12}$
Prior believe under bias
Nominal cumulative value
π_{bias}
Observed reward, $\gamma_{\epsilon 1}$
t+1
t+2
t+3
t+12
Time

Figure 4.
Leader's prior believe is biased and this strategy leads to delayed punishment.

Leader reward function is (γ_L) the combination of monthly profit change, and expected affect to future profit. π_{Li} is the leader's strategy at current state (month). It seems that at the beginning, the leader strategy is weighted at the monthly profit and the expected future reward is based on simple linear regression of data achieved so far. This means biased prior believe where the expected reward is not nearly the same as the outcome of the strategy. Thus, the value function under biased strategy is the following:

$$\gamma_L = \gamma_{t€} + \gamma_{12€} \tag{4}$$

where

$\gamma_{t€}$ is the observed state reward.

$\gamma_{12€}$ is the expected future reward.

When the leader gets more experience and learns to understand the complexity of the system as well as the meaning of workers' QWL, the prior believe value function changes. QWL change starts to be more interesting, because leader learn to expect more future profit when QWL improves. Thus, along this information the leader adjusts the strategy for optimizing cumulative yearly profit. Here the leadership game stochastic nature is key to learning the Nash general sum equilibrium between the QWL and profit.

$$\gamma_L = \alpha_t \left(\gamma_{t€} + \gamma_{12€}\right) \tag{5}$$

where

$\gamma_{t€}$ is the observed state reward.

$\gamma_{12€}$ is the expected future reward by improving the QWL.

α_t is the learning rate.

QWL is improved by leadership actions that reduce the monthly working time for making the revenue. Thus, improving QWL reduces monthly revenue and profit, but may increase effective working time in the future and so increase the future profit. In monthly basis, this phenomenon may be contradictory and confusing, but by practice, the best reward is achieved where both workers' and leader's payoff functions flourish. This means the Nash equilibrium where yearly QWL is improved with high profit. In Nash equilibrium, leader's choices are the best response to the workers' signals and business cumulative outcome at the end of the year.

Bayesian stochastic strategic non-symmetric signaling learning game follows Markov decision process [20–22]. Management-game forms stochastic game tuple

$$[N,S,C,A,T,P,R] \tag{6}$$

where

N is set of players, i.

S is set of states, s.

C is set of competences at actions a.

A is set of actions, a.

T is set of signals, τ.

P is transition probability function; P: S x A x C thus P(s, c, a), $\rho{:}SxA|C \rightarrow \Delta$ is the transition function, where Δ is the set of probability distributions over state space S.

R is reward function, R = r1,...rn, $\gamma{:}SxA|C \rightarrow R$.

There is incomplete but perfect information. The agents (workers and leader) do not know other agents' payoff functions in detail, but they can observe other agents' immediate payoffs and actions from past months. A leader does not know exactly which actions would be the best but can choose actions that should be good enough. The leader will get workers emotional feedback immediately and information from profit monthly change and cumulative reward. After several game rounds, the player (leader) will learn the optimal actions to improve both the QWL and annual profit. Thus, the player will achieve the Nash equilibrium of stochastic Markov learning game.

5. Management-game Markov sequences

Management-game has context specific Markov-sequences. State and state change transition follows the Markov property where the future is independent of the past given the current situation. Once a state is defined, its change is determined by the behavior of the parties. State change is sequential, following the players actions and state transition probability function. Sequences are:

First Month (January)

1. Workers interpret the state situation and give signals based on prior believe (τ).
2. Leader observes the signals and updates the signal-strategy (π_τ).
3. Leader updates standard-strategy (π_{st}). Note: at this first month there is no data to update this year profit strategy.
4. Leader makes actions (or decide doing nothing) (a)
5. Actions leads to state change with possible outside intervention (stochastic)
6. Leader observes immediate ($\gamma_{€1}$) and cumulative ($\gamma_{\Sigma €}$) profit rewards (or sacrifices). From now on, the leader gets also profit outcome, thus updates also profit strategy.
7. According the combination of rewards, the leader upgrades prior believes concerning own behavior
8. Leader upgrades profit-strategy and standard-strategy for choosing actions t + 1
9. Workers give signals to be considered when deciding actions t + 1
10. Leader updates signal-strategy for choosing actions t + 1

11–13. Leader makes actions t + 1 in line with all three strategies.

14. State transition to state t + 1

15... From now on, the supervisor should update all three strategies simultaneously as learning sequences progress (**Figures 5** and **6**).

Leadership game Q-learning function is (7)

$$Q_{t+1}(s,c,a) = (1-\alpha)Q_t(s,c,a) + \alpha\left[\gamma_{\Delta i} + \gamma_{\Delta 12} + \beta \max_a Q_t(s_{t+1}, c, a)\right] \tag{7}$$

where
β is [0,1] is discounted reward factor
α_t is [0,1] is the learning rate $(1-\alpha_t)$
$\gamma_{\Delta i}$ is the monthly profit reward
$\gamma_{\Delta 12}$ is the expected cumulative profit reward (floating 12 months)
$\max_a Q_t(s_{t+1},,c,,a)$ is expected maximum equilibrium strategy state change pay from best actions a at competence levels c.

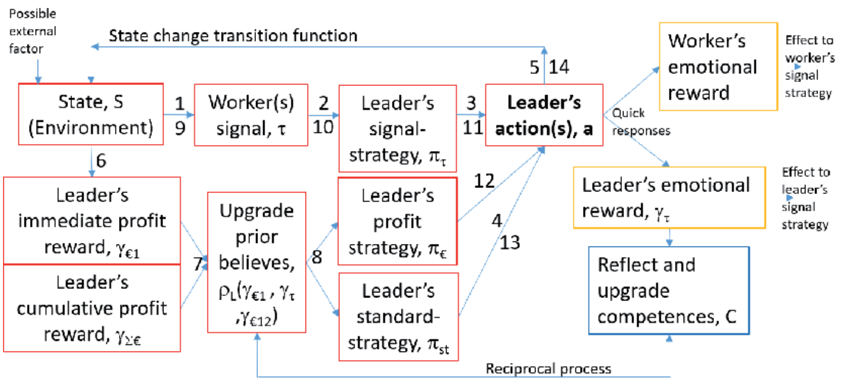

Possible external factor
State change transition function
State, S (Environment)
1
9
Worker(s) signal, τ
2
10
Leader's signal-strategy, π_τ
3
11
Leader's action(s), a
5
14
Worker's emotional reward
Effect to worker's signal strategy
Quick responses
6
Leader's immediate profit reward, $\gamma_{\epsilon 1}$
Leader's cumulative profit reward, $\gamma_{\Sigma\epsilon}$
7
Upgrade prior believes, $\rho_L(\gamma_{\epsilon 1}, \gamma_\tau, \gamma_{\epsilon 12})$
8
Leader's profit strategy, π_ϵ
Leader's standard-strategy, π_{st}
12
4
13
Leader's emotional reward, γ_τ
Effect to leader's signal strategy
Reflect and upgrade competences, C
Reciprocal process

Figure 5.
Management-game Markov sequences.

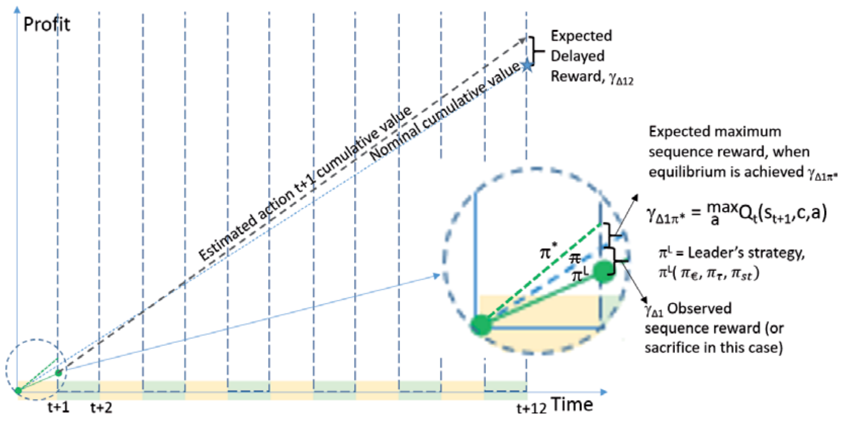

Profit
Expected Delayed Reward, $\gamma_{\Delta 12}$
Estimated action t+1 cumulative value
Nominal cumulative value
Expected maximum sequence reward, when equilibrium is achieved $\gamma_{\Delta 1\pi^*}$
$\gamma_{\Delta 1\pi^*} = \max_a Q_t(s_{t+1},c,a)$
π^*
π
π^L
π^L = Leader's strategy, $\pi^L(\pi_\epsilon, \pi_\tau, \pi_{st})$
$\gamma_{\Delta 1}$ Observed sequence reward (or sacrifice in this case)
t+1
t+2
t+12
Time

Figure 6.
Management game learning phenomenon for finding equilibrium.

With expected equilibrium strategy pay ($\max_a Q_t(s_{t+1},,c,,a)$) you can calibrate the Q-learning points so that it gives approximately 0-points when no actions are done, thus no learning was achieved. In our Q-learning function this value is 221 € monthly improvement value per employee. This corresponds the costs of one absence day per month for each worker or one working day more in work efficiency. Using this value, Q-learning gives 0 points regardless of what the supervisor's skills are.

6. Management learning strategies

There are three different strategic areas of prior-believes that forms the manager's learning context. These strategies are influenced by the supervisor's interaction skills (competences), which tend to either promote or hinder learning in the area. Every manager has personal competences, which seems to form personal Nash equilibrium and corresponding Q-learning results. According to this article, it seems that Nash equilibrium is different for each combination of manager's competences. In addition, the leader's strategic mind-set defines the equilibrium. Indeed, management equilibrium seems to be evolving phenomenon, depending on organization and its' players change of characteristics (**Figure** 7).

The focus of the signal-strategy ($\pi_{\hat{o}}$) is to learn to understand employee signals and utilize them to achieve best reward. This strategy is strongly related to the psychological agreement between workers and supervisor. When working team members learn to play general-sum-game, the signals are provided early and in constructive way, which foster optimal actions. In case signal-strategy turn to 0-sum-game the signals tend to be hided or used to harm other members of the team. Thus, creating best foundation for signal-strategy is grounded on continuous fostering of psychological agreement at the working society.

Profit-strategy ($\pi_{€}$) focus is to learn from experience how target profit is achieved at anticipated time span. Economical profit indicators are usually constantly monitored, giving them a lot of attention. In addition, organization profit target time span is determined at management system, which create certain predefined attitude towards achieving profit. From a strategic point of view, there is a big difference between focusing on maximum result this month or aiming for the maximum profit with delay of several months. If a management system requires maximum results over a short period of time, then it reinforces the detrimental profit-maximization bias. In this bias the team-leader tend to push workers performance too much, which lead to maximizing performance that is declining. In addition, a manager under this bias neglect employee signals because the signals pose a risk that short-term profits are threatened when scarce working hours are used to solve the problem. Clearly, this behavior damages the signal game, as employees learn that problems are not worth reporting.

The focus of the standard-strategy (π_{st}) is to learn how to plan actions in advance to secure the reward in the future. Usually this strategy comes from the

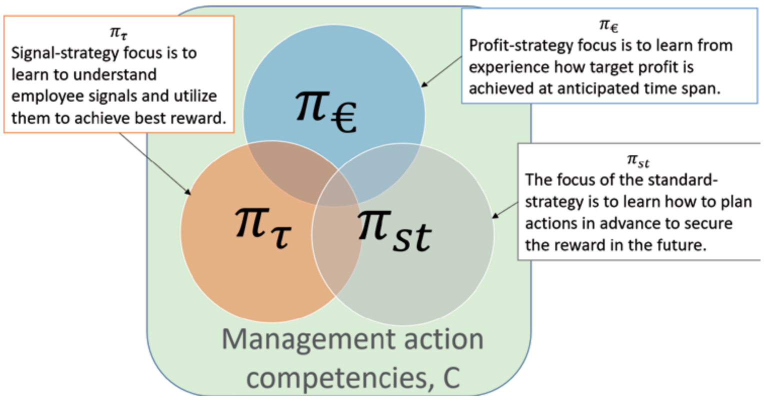

π_τ
Signal-strategy focus is to learn to understand employee signals and utilize them to achieve best reward.
$\pi_€$
Profit-strategy focus is to learn from experience how target profit is achieved at anticipated time span.
$\pi_€$
π_τ
π_{st}
π_{st}
The focus of the standard-strategy is to learn how to plan actions in advance to secure the reward in the future.
Management action competencies, C

Figure 7.
Management learning strategies.

organization's human resources management, which recommends the implementation of certain management practices according to the annual plan. In practice it is common that this recommended plan is followed in various ways – some managers follow the plan while others do not. Those who do not follow the plan are likely to have learned good reasons why the recommended measures are not be implemented. Approved defense excuses may be related to the lack of time, because profit target needed all the focus. Clearly, this behavior damages the benefits of good standard-strategy.

All of these supervisor strategies are built on the supervisor's personal and ever-evolving managerial skills. In this management game theoretical approach there are personal leadership action competencies that determine the effect of each action. There is interaction between management competencies and learning strategies. The supervisor reflects the effectiveness of his or her own leadership behavior and changes personal management strategies accordingly.

7. Digital twin AI advisor using Bellman function

Digital twin advisor uses Bellman [20] expectation function in finding optimal actions for achieving Nash equilibrium. Bellman expectation function for strategy π is

$$v_{\eth}(s) = E_{\pi}\left[R_{t+1} + \beta v(S_{t+12})\right] \tag{8}$$

where

R_{t+1} = immediate reward

$\beta v(S_{t+12})$ = discounted future value (12 months estimation).

Optimal policy forms from the actions that result in optimal value function, thus

$$q_{\eth *}(s,,c,,a) = R^{s}_{\eth *} + \beta^{arcmax}_{a\in A} v_{\pi *}(S_{t+12}) \tag{9}$$

where

$R^{s}_{\eth *}$ = immediate state reward from strategy π*

$\beta^{arcmax}_{a\in A} v_{\pi *}(S_{t+12})$ = discounted maximum future value (12 months estimation).

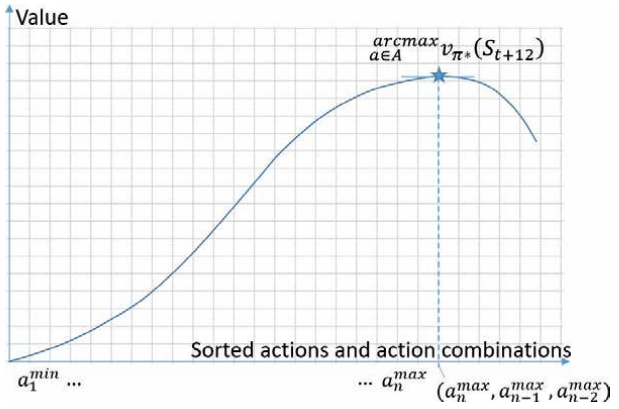


Figure 8.
Bellman function principle of marginal productivity value.

In our digital twin AI assistant is using Bellman function. It returns the combination of actions that gives the best value after floating 12 months. This is achieved so that first each action value is analyzed and sorted in magnitude of the value. Then the combinations of best actions are evaluated until marginal productivity of the value is achieved, see example at **Figure 8**. One simulation episode is 12 months; thus, the Bellman function maximize future reward even when the episode is coming to end.

Simulation game is done using UNITY 3D, for making possible to play the learning game episodes. Each episode is 12 months, consisting several workplace challenges. In the test runs we used Cash Cow episode where problems are easy, the market situation is steady, and the company does not seek special increase in revenue. State space problems are signaled by the workers that comes meeting the team-leader (agent). In this ODT there is so far 25 workplace challenges which reduce QWL according situational probability matrix. Leader has 32 best management practices (action space) that may be used as the leader prefers. Each action reduce profit and may improve QWL according state space situation specific probability function [23] (**Figure 9**).

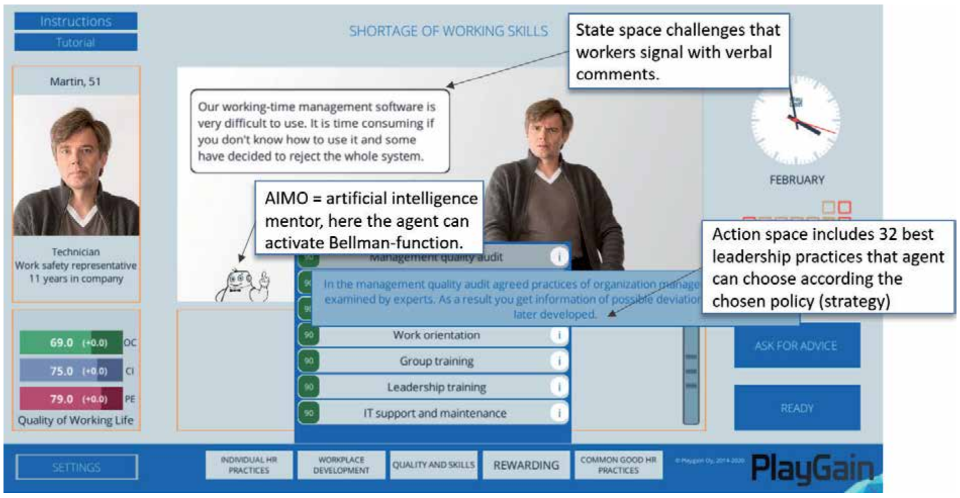
We tested simulation using three different competence values; 30%, 60% and 90%. **Table 1** contains the results of three simulation rounds as follows:

- BIAS = human simulation episode (round) with bias to maximize short term profit. Only problem-solving actions are made. In BIAS episode the focus is on maximizing short-term profit.

- Learning = human simulation episode where leader has learned to maximize best result in QWL and profit. Agent execute best learning strategy (see **Figure 7**) with long-term profit mind-set, problems solving as good as possible and following yearly management-plan of actions.

- Bellman = artificial intelligence episode where all actions are chosen according Bellman function (see **Figure 8**).

It seems that with management competence levels 30% there are difficulties to achieve budgeted target result in profit. If QWL is sacrificed for short term wins, the cumulative profit result at the end of the year will be poor. It seems that in


SHORTAGE OF WORKING SKILLS

State space challenges that workers signal with verbal comments.

Our working-time management software is very difficult to use. It is time consuming if you don't know how to use it and some have decided to reject the whole system.

AIMO = artificial intelligence mentor, here the agent can activate Bellman-function.

Action space includes 32 best leadership practices that agent can choose according the chosen policy (strategy)

In the management quality audit agreed practices of organization manage examined by experts. As a result you get information of possible deviatio later developed.

Martin, 51
Technician
Work safety representative
11 years in company

69.0	(+0.0)	OC
75.0	(+0.0)	CI
79.0	(+0.0)	PE

Quality of Working Life

Figure 9.
Simulation game user-interface.

	Q-learning	QWL start	QWL end	QWL difference	Cumulative		Profit difference	Equilibrium in 1 y.
					Budj. €	EBITDA		
Competence 30%, BIAS	3 310	60,2%	57,9%	-2,3%	254 923	244 921	−10 002	—
Competence 30%. Learning	5 370	60,2%	64,6%	4,4%	254 923	243 650	−11 273	yes
Competence 30%, Bellman	21412	60,2%	67,5%	7,3%	254 923	257 070	2 147	yes
Competence 60%, BIAS	5 854	60,2%	59,0%	−1,2%	254 923	263 284	8 361	yes
Competence 60%, Learning	20 425	60,2%	68,3%	8,1%	254 923	287 083	32 160	yes
Competence 60%, Bellman	35 931	60,2%	70,4%	10,2%	254 923	293 442	38 519	no
Competence 90%, BIAS	7 737	60,2%	59,8%	−0,4%	254 923	276 828	21 905	yes
Competence 90%, learning	31 240	60,2%	69,9%	9,7%	254 923	305 604	50 681	no
Competence 90%, Bellman	38 446	60,2%	70,1%	9,9%	254 923	312 003	57 080	no

Table 1.
Test episode values.

one-year simulation episode there is achieved equilibrium where Q-learning points and QWL values are not exceeding. At 30% competence levels the BIAS episode Q-learning points varies between 0 and 3000 points. It seems as if the agent has no idea of how to achieve sustainable development where both QWL and profit improves. With low competence levels only with Bellman decisions will the profit slightly exceed the target value.

Manager's competence levels 60% are quite realistic, representing average line-managers leadership-action skills. In one-year simulation both BIAS and Learning strategies achieve Nash equilibrium, however in different profit outcome. At BIAS strategy the QWL is set at level 60%, which actually corresponds workforce medium QWL value in Finland [24]. When equilibrium is achieved, it may be difficult to change the behavior (see **Figure 10**).

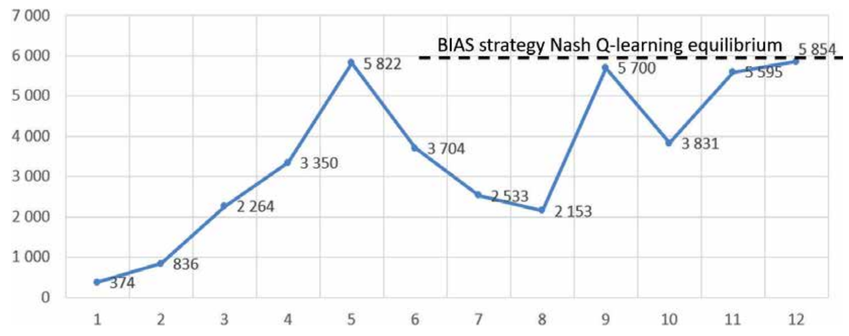


Figure 10.
BIAS strategy Q-learning points.

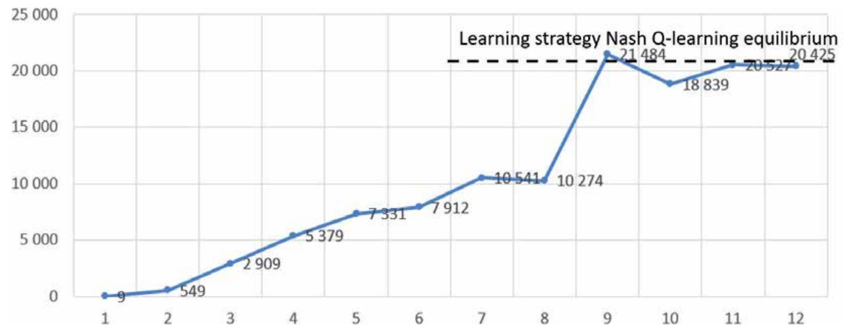


Figure 11.
Learning strategy Nash Q-learning equilibrium.

Learning strategy has also equilibrium at competence level 60%, but higher QWL and profit values than in BIAS (see **Figure 11**). In our practical simulation studies this type of results are usually learned when simulation episodes are practiced over ten times. Must bear in mind that management systems have tendency to press maximizing short-term profits, thus remaining in BIAS mind set. Learning to be excellent leader requires several years practice in organization system that allows investing in people. This phenomenon may explain why some leaders learn to be excellent team-leaders while majority remains at lower level.

There is interesting phenomenon at 90% competence level BIAS strategy. Even with very high leadership skills the QWL is set at 60% where equilibrium remains. This is due to the behavior where leadership actions are implemented only when problems arise, thus there are no proactive investments in team development. In competence levels 90% it seems that one-year simulation episode is not enough time to achieve perfect equilibrium at Learning and Bellman strategies, since Q-learning points and QWL seems to continue improving throughout the episode. It would need longer time period to achieve equilibrium.

BIAS strategy seems to achieve equilibrium where QWL is no longer improved and the Q-learning points finds management cultural maximum value. The lower the competence, the lower the level of QWL, however the difference is not so big, varying from 57% to 60%. This is interesting because in Finland the workforce medium QWL is around 60% [24]. One could argue that the profit maximization bias is common and not depending on line-managers leadership competences, and therefore most employees feel the QWL is around 60%. Moreover, the reason for profit maximization bias is not necessarily a lack of leaders' skills, but a management system that forces leaders to focus on short-term profit rather than people.

8. Conclusions and discussion

Organizational management research has typically focused on qualitative behavioral factors that have a complex relationship to organizational success, and in addition, impacts often come with a delay. Each organization is a unique system with certain same laws, but also a unique context of its own. Therefore, repeating the empirical research results has proven to be challenging, which also makes it difficult to draw generalizable conclusions [7]. This article examines the utilization of model-based artificial intelligence in management development. ODT can be used to assess the impact of management behavior on an organization's success, considering situational data and the impact of management culture. ODT helps to explore the fundamental nature of an organization, which means a metaphysical essence in where everything affects everything.

The article uses artificial intelligence to illustrate how leadership behavior can create a so-called QWL glass roof that invisibly prevents teams from growing to the top performing category. The management system forms the behavior of supervisors in such a way that harmful biases of management thinking may occur, in which case people's performance does not develop favorably. These harmful biases of thought are very complex as they include phenomenon of delayed effects on an organization's competitiveness. Model-driven reinforcement learning artificial intelligence reveals a variety of human and complex mechanisms that hinder the development of competitiveness.

Reinforcement learning is following rational learning phenomenon, where learning take place gradually, according the experience. Simulation model provides learning platform where person can learn without fear of remorse. This is essential especially for managers, because in real life there is hardly room for learning from mistakes. The ODT models the situation with the organization's own data. The simulation can be designed according to the company's own strategy, allowing future challenges to be practiced. This allows management and supervisors to adapt in advance and prepare for future challenges. More proactive management reduces the realization of personnel and business risks and adds value to performance. For example, adapting to a recession can be practiced, as can market growth, both of which require a different way of managing. Artificial intelligence combined with the digital twin helps to emphasize leadership skills and practices that lead to sustainable development.

ODT has been used in college students' leadership studies. Learning outcomes have been monitored through self-assessments, and the results are encouraging. Gamified simulation-learning is based on reinforcement learning, where progress takes place through experiential adaptation according to the student's capabilities and learning ability [25]. ODT is also used in managerial trainings for companies and municipal organizations. Perhaps the biggest challenge in coaching supervisors in working life is unlearning the biases that prevent leadership success. Traditional teaching is largely based on sharing best knowledge, where the teacher shares information on how to act and why to behave in a certain way. The power of digital simulation teaching is based on the fact that it adapts the brain through experiential learning. When a supervisor has to change the prevailing leadership attitude, he or she kind of adapts the brain to another frequency where listening and caring for employees rises higher in priorities. In this way, the supervisor becomes interested in developing herself in interaction practices where she may not have previously felt the need to learn.

The architecture of the digital twin models the reality of an organization with relatively good accuracy, which is important in building trust in an artificial intelligence solution. The core of the model is in the Human Capital Production Function of and in the scientific research of the Quality of Working Life index [26]. The architecture lays the foundation for a neural network that has been fine-tuned with the probabilities of empirical research as well as correlations created through supervised learning. For example, the physical and emotional safety (PE) of the QWL index correlates with sickness absence, so that when the PE factor falls, sick leaves increases. The correlation is brought into the digital twin, which makes the model more accurate because it also models sick leaves. In addition to research data, the digital twin can be calibrated with data from the organization. ODT learning can be extended in the organizational hierarchy to the level of an individual supervisor. In this way, artificial intelligence learns the strengths and weaknesses of a leader, so that the advice given by artificial intelligence is targeted at each supervisor.

Supervised learning AI that is based on data alone is unable to "understand" organizational complexity and phenomenon of delayed impact relationships. In fact, there is a word of warning in using simple data-driven AI in complex organization environment, because it may strengthen the harmful behavioral biases. Article indicates that ODT with Bellman algorithm can be used in finding organization specific optimal behavioral patterns and measures which will form sustainable competitiveness. The article suggests that in the future, top-tier companies will use RL artificial intelligence to support management decision-making.

Author details

Marko Kesti
University of Lapland, Rovaniemi, Finland

*Address all correspondence to: marko.kesti@ulapland.fi

References

[1] Bersin J., Geller J., Wakefield N. and Walsh B. (2016). The new organization: Different by design. In Global Human Capital Trends study available at https://www2.deloitte.com/us/en/insights/focus/human-capital-trends/2016/human-capital-trends-introduction.html

[2] Kesti, M. (2012). The tacit signal method in human competence-based organization performance development. Rovaniemi: Lapland University Press.

[3] Pfeffer, J. (1994). Competitive advantage through people: Unleashing the power of the workforce. Boston: Harvard University Press.

[4] Ulrich, D. (1997). Human resource champions, the next agenda for adding value and delivering results. Boston: Harvard Business School Press.

[5] Guest, D. (1997). Human Resource Management and Performance: A Review and Research Agenda. International Journal of Human Resource Management, 8(3), 263-276.

[6] Becker B., & Huselid M. (2006). Strategic human resources management: Where do we go from here? Journal of Management, 32, 898-925.

[7] Fleetwood S., & Hesketh, A. (2010). Explaining the performance of human resource management. Cambridge University Press. Wright & Bonett, 2007.

[8] Wright, T. A., & Cropanzano, R. (2004). The role of psychological well-being in job performance: A fresh look at an age-old quest. Organizational Dynamics, 33(4), 338-351.

[9] Bourne Vanson (2018). Future of Work study of 4,600 business leaders from companies with 250+ employees, across 40+ countries and 12 industries, Dell Technologies. Available at https://www.delltechnologies.com/en-us/perspectives/future-of-work.htm

[10] Natarajan Balasubramanian, Yang Ye and Mingtao Xu (2020). Substituting Human Decision-Making with Machine Learning: Implications for Organizational Learning, Academy of Management Review VOL. 0.

[11] DellTechnologies (2020). The Future of Digital Twins at https://www.delltechnologies.com/en-us/perspectives/the-future-of-digital-twins/, December 1, 2020

[12] Kesti, M., Leinonen, J. and Syväjärvi, A. (2016). A Multidisciplinary Critical Approach to Measure and Analyze Human Capital Productivity. In Russ, M. (ed.). Quantitative Multidisciplinary Approaches in Human Capital and Asset Management (pp 1-317). Hershey, PA: IGI Global. (1-22).

[13] Kesti, M. and Syväjärvi, A. (2015). Human Capital Production Function in Strategic Management. Technology and Investment, 6, 12-21.

[14] Nash J. (1951). Non-Cooperative Games. The Annals of Mathematics, Second Series, Vol. 54, No. 2, pp. 286-295.

[15] Alderfer, C. P. (1967). Convergent and discriminant validation of satisfaction and desire measures by interviews and questionnaires. Journal of Applied Psychology, 51(6), 509-520.

[16] Antonovsky A. (1979). Health, stress and coping. San Francisco: Jossey-Bass.

[17] Kano, N. (1984). Attractive quality and must-be quality. Journal of the Japanese Society for Quality Control, April, 39-48.

[18] Herzberg, F., Mausner, B. and Snyderman, B., B. (1959). The motivation to work, Second edition, John Wiley & Sons, New York.

[19] Watkins C. and Dayan P. (1992). Q-learning. Machine learning, 3: 279-292.

[20] Bellman, R. (1957). "A Markovian decision process". Journal of mathematics and mechanics. 6(5), 679-684.

[21] Littman M.L. (1994). Markov games as a framework for multi-agent reinforcement learning, Proc. 11th international conference on machine learning, New Brunswick, pp. 157-163.

[22] Sutton R. S. and Barto A. G. (1998) Reinforcement learning: an introduction, MIT Press/Bradford Books.

[23] Leadermind simulation learning game introduction. (2020). Playgain Inc. Available at www.leadermind.fi

[24] Savukoski T. (2017). Työelämän laadun indeksin yhteys työkyvyttö-myyseläkeriskiin. ProGradu, Lapin Yliopisto.

[25] Kesti, M. , Ylitalo, A.-I. and Vakkala H. (2019). Management Game: Gamifying Leadership Learning, International Journal of Innovation in the Digital Economy, Volume 10, Issue3.

[26] Kesti M. (2019) Architecture of Management Game for Reinforced Deep Learning. In: Arai K., Kapoor S., Bhatia R. (eds) Intelligent Systems and Applications. IntelliSys 2018. Advances in Intelligent Systems and Computing, vol 868. Springer, Cham.

Chapter 4

Deep Learning for Subtyping and Prediction of Diseases: Long-Short Term Memory

Hayrettin Okut

Abstract

The long short-term memory neural network (LSTM) is a type of recurrent neural network (RNN). During the training of RNN architecture, sequential information is used and travels through the neural network from input vector to the output neurons, while the error is calculated and propagated back through the network to update the network parameters. Information in these networks incorporates loops into the hidden layer. Loops allow information to flow multi-directionally so that the hidden state signifies past information held at a given time step. Consequently, the output is dependent on the previous predictions which are already known. However, RNNs have limited capacity to bridge more than a certain number of steps. Mainly this is due to the vanishing of gradients which causes the predictions to capture the short-term dependencies as information from earlier steps decays. As more layers in RNN containing activation functions are added, the gradient of the loss function approaches zero. The LSTM neural networks (LSTM-ANNs) enable learning long-term dependencies. LSTM introduces a memory unit and gate mechanism to enable capture of the long dependencies in a sequence. Therefore, LSTM networks can selectively remember or forget information and are capable of learn thousands timesteps by structures called cell states and three gates.

Keywords: deep learning, recurrent neural networks, long-short term memory

1. Introduction

Artificial neural networks (ANNs) are a type of the computing system that mimics and simulates the function of the human brain to analyze and process the complex data in an adaptive approach. They are capable of implementing massively parallel computations for mapping, function approximation, classification, and pattern recognition processing that require less formal statistical training. Moreover, ANNs have the ability to identify very high complex nonlinear relationships between outcome (dependent) and predictor (independent) variables using multiple training algorithms [1, 2]. Generally speaking, in terms of network architectures, ANNs tend be classified into two different classes, feedforward and recurrent ANNs, each may have several subclasses.

Feedforward is a widely used ANN paradigm for, classification, function approximation, mapping and pattern recognition. Each layer is fully connected to neurons in another layer with no connection between neurons in the same layer.

As the name suggests, information is fed in a forward direction from the input to the output layer through one or more hidden layers. MLP feed forward with one hidden layer can virtually predict any linear or non-linear model to any degree of accuracy, assuming that you have a appropriate number of neurons in hidden layer and an appropriate amount of data. Adding more neurons in the hidden layers to an ANN architecture gives the model the flexibility of fitting extremely complex nonlinear functions. This also holds true in classification modeling in approximating any nonlinear decision boundary with great accuracy [2].

Recurrent neural networks (RNNs) emerged as an operative and scalable ANN model for several learning problems associated with sequential data. Information in these networks incorporate loops into the hidden layer. These loops allow information to flow multi-directionally so that the hidden state signifies past information held at a given time step. Thus, these network types have an infinite dynamic response to sequential data. Many applications, such as Apple's Siri and Google's voice search, use RNN.

The most popular way to train a neural network is by backpropagation (BP). This method can be used with either feedforward or recurrent networks. BP involves working backward through each timestep to calculate prediction errors and estimate a gradient, which in turn is used to update the weights in the network. For example, to enable the long sequences found in RNNs, multiple timesteps are conducted that unrolls the network, adds new layers, and recalculates the prediction error, resulting in a very deep network.

However, standard and deep RNN neural networks may suffer from vanishing or exploding gradients problems. As more layers in RNN containing activation functions are added, the gradient of the loss function approaches zero. As more layers of activating functions are added, the gradient loss function may approach zero (vanish), leaving the functions unchanged. This stops further training and ends the procedure prematurely. As such, parameters capture only short-term dependencies, while information from earlier time steps may be forgotten. Thus, the model converges on a poor solution. Because error gradients can be unstable, the reverse issue, exploding gradients may occur. This causes errors to grow drastically within each time step (MATLAB, 2020b). Therefore, backpropagation may be limited by the number of timesteps.

Long Short-Term Memory (LSTM) networks address the issue of vanishing/exploding gradients and was first introduced by [3]. In addition to the hidden state in RNN, an LSTM block includes memory cells (that store previous information) and introduces a series of gates, called input, output, and forget gates. These gates allow for additional adjustments (to account for nonlinearity) and prevents errors from vanishing or exploding. The result is a more accurate predicted outcome, the solution does not stop prematurely, nor is previous information lost [4].

Practical applications of LSTM have been published in medical journals. For example, studies [5–14] used different variants of RNN for classification and prediction purposes from medical records. Important, LSTM does not have any assumption about elapsed time measures can be utilized to subtype patients or diseases. In one such study, LSTM was used to make risk predictions of disease progression for patients with Parkinson's by leveraging longitudinal medical records with irregular time intervals [15].

The purpose of this chapter is to introduce Long Short-Term Memory (LSTM) networks. This chapter begins with introduction of multilayer feedforward architectures. The core characteristic of an RNN and vanishing of the gradient will be explained briefly. Next, the LSTM neural network, optimization of network parameters, and the methodology for avoiding the vanishing gradient problem will be covered, respectively. The chapter ends with a MATLAB example for LSTM.

2. Artificial neural networks and multilayer neural network

Artificial Neural Networks (ANNs) are powerful computing techniques that mimic functions of the human brain to solve complex problems arising from big and messy data. As a machine learning method, ANNs can act as universal approximators of complex functions capable of capturing nonlinear relationships between inputs and outcomes. They adaptively learn functional structures by simultaneously utilizing a series of nonlinear and linear activation functions. ANNs offer several advantages. They require less formal statistical training, have the ability to detect all possible interactions between input variables, and include training algorithms adapted from backpropagation algorithms to improve the predictive ability of the model [2].

Feed forward multilayer perceptron (**Figure 1**) is the most used in ANN architectures. Uses include function approximation, classification, and pattern recognition. Similar to information processing within the human brain, connections are formed from successive layers. They are fully connected because neurons in each layer are connected to the neurons from the previous and the subsequent layer through adaptable synaptic network parameters. The first layer of multi-layer ANN is called the input layer (or left-most layer) that accepts the training data from sources external to the network. The last layer (or rightmost layer) is called the output layer that contains output units of the network. Depending on prediction or classification, the number of neurons in the output layer may consist of one or more neurons. The layer(s) between input and output layers are called hidden layer(s). Depending on the architecture multiple hidden layers can be placed between input and output layers.

Training occurs at the neuronal level in the hidden and output layers by updating synaptic strengths, eliminating some, and building new synapses. The central idea is to distribute the error function across the hidden layers, corresponding to their effect on the output. **Figure 1** demonstrates the architecture

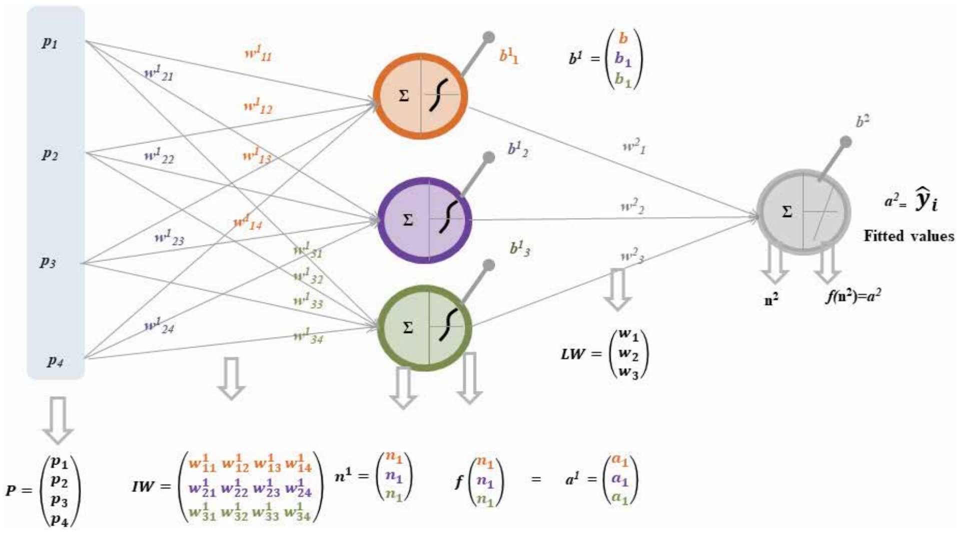

p_1
p_2
p_3
p_4
w^1_{11} w^1_{12} w^1_{13} w^1_{14}
w^1_{21} w^1_{22} w^1_{23} w^1_{24}
w^1_{31} w^1_{32} w^1_{33} w^1_{34}
b^1_1 b^1_2 b^1_3
$b^1 = (b, b_1, b_1)$
b^2
w^2_1 w^2_2 w^2_3
$a^2 = \hat{y}_i$
Fitted values
n^2
$f(n^2)=a^2$
$LW = (w_1, w_2, w_3)$
$P = (p_1, p_2, p_3, p_4)$
IW
$n^1 = (n_1, n_1, n_1)$
$f(n_1, n_1, n_1) = a^1 = (a_1, a_1, a_1)$

Figure 1.
(adapted from Okut, 2016). Artificial neural network design with 4 inputs (p_i)*. Each input is connected to up to 3 neurons via coefficients* $w_{kj}^{(l)}$ (l *denotes layer;* j *denotes neuron;* k *denotes input variable). Each hidden and output neuron has a bias parameter* b_j^l*. Here P = inputs,* ***IW*** *= weights from input to hidden layer (12 weights),* ***LW*** *= weights from hidden to output layer (3 weights),* b^1 *= Hidden layer biases (3 biases),* b^2 *= Output layer biases (1 bias),* $n^1 = \boldsymbol{IWP} + b^1$ *is the weighted summation of the first layer,* $a^1 = f(n^1)$ *is output of hidden layer,* $n^2 = \boldsymbol{LW}a^1 + b\ 2$ *is weighted summation of the second layer and* $\hat{t} = a^2 = f(n^2)$ *is the predicted value of the network. The total number of parameters for this ANN is 12 + 3 + 3 + 1 = 19.*

of a simple feedforward MLP where P identifies the input layer, next one or more hidden layers, which is followed by the output layer containing the fitted values. The feed forward MLP networks is evaluated in two stages. First, in the feedforward stage information comes from the left and each unit evaluates its activation function$f$. The results (output) are transmitted to the units connected to the right. The second stage involves the backpropagation (BP) step and is used for training the neural network using gradient descent algorithm in which the network parameters are moved along the negative of the gradient of the performance function The process consists of running the whole network backward and adjusting the weights (and error) in the hidden layer. The feedforward and backward steps are repeated several times, called epochs. The *algorithm stops when the value of the loss (error) function has become sufficiently small*.

3. Recurrent neural networks

Because the two stages, outlined in **Figure 1** above, are used in all neural networks, we can extend MLP feedforward neural networks for use in sequential data or time series data. To do so, we must address time sequences. Recurrent neural networks (RNNs) address this issue by conducting multiple time steps that unrolls the network, adds new layers, and recalculates the prediction error, resulting in a very deep network. First, the connections between nodes in the hidden layer(s) form a directed graph along a temporal sequence allowing information to persist. Through such a mechanism, the concept of time creates the RNNs memory. Here, the architecture receives information from multiple previous layers of the network.

Figure 2 outlines the hidden layer for RNN and demonstrates the nonlinear function of the previous layers and the current input (p). Here, the hyperbolic tangent activation function is used to generate the hidden state. The model has memory since the bias term is based on the "past". As a consequence, the outputs from the previous step are fed as input to the current step. Another way to think about RNNs is that a recurrent neural network has multiple copies of the same network, each passing a message to a successor (**Figure 3**). Thus, the output value of the last time point is transmitted back to the neural network, so that the parameter estimation (weight calculation) of each time point is related to the content of the previous time point.

3.1 Training recurrent neural networks

Similar to feedforward MLP networks, RNNs have two stages, a forward and a backward stage. Each works together during the training of the network. However, structures and calculation patterns differ. Let us first consider the forward pass.

Stage 1*: Forward pass.*

The forward pass will be summarized into 5 steps:

1. Summation step. In this step two different source of information are combined before nonlinear activation function will be take place. The sources are the values of weighted input $\left(\boldsymbol{W}^{(\boldsymbol{p})}\boldsymbol{p}_t\right)$ and weighted previous hidden state with bias $\left(\boldsymbol{h}_{t-1}\boldsymbol{W}^{(\boldsymbol{h})} + \boldsymbol{b}^{(\boldsymbol{h})}\right)$. Here, $\boldsymbol{p}_t$ and $\boldsymbol{W}^{(\boldsymbol{p})}$ are input vector and the input weight matrix, $\boldsymbol{h}_{t\text{-}1}$ is value of previous hidden state, $\boldsymbol{W}^{(\boldsymbol{h})}$ is weight matrix of hidden state pertains the previous hidden state with the current one and $\boldsymbol{b}^{(\boldsymbol{h})}$. Is bias. Since the previous hidden state and current input are measured as

vectors, each element in the vector is placed in a different orthogonal dimension

$$a_h(t) = \left(W^{(p)}p_t + h_{t-1}W^{(h)} + b^{(h)}\right) \tag{1}$$

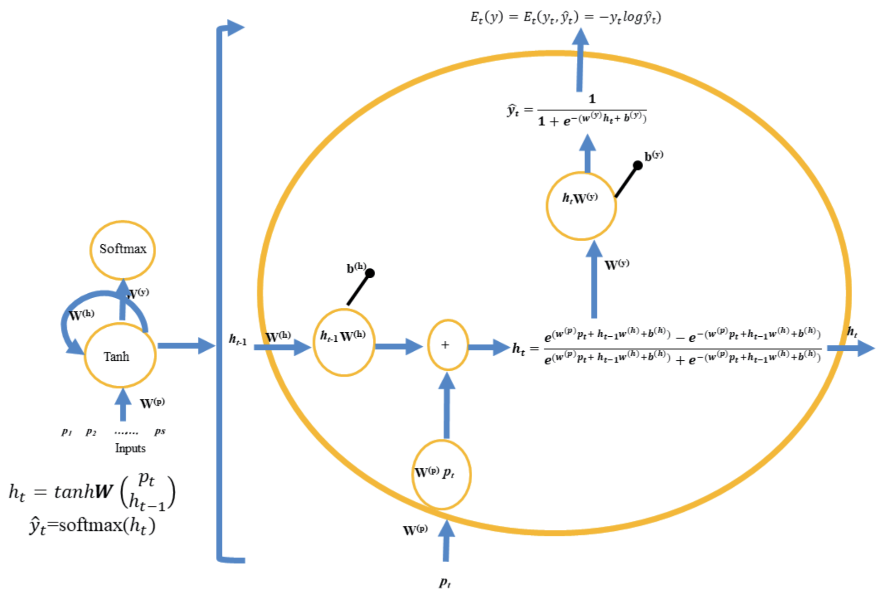

$E_t(y) = E_t(y_t, \hat{y}_t) = -y_t log\hat{y}_t)$
$\hat{y}_t = \frac{1}{1+e^{-(w^{(y)}h_t + b^{(y)})}}$
$b^{(y)}$
$h_t W^{(y)}$
$W^{(y)}$
Softmax
$W^{(h)}$
Tanh
$W^{(p)}$
p_1 p_2 .... ps
Inputs
$h_t = tanh\mathbf{W}\begin{pmatrix}p_t\\h_{t-1}\end{pmatrix}$
$\hat{y}_t$=softmax(h_t)
h_{t-1}
$b^{(h)}$
$h_{t-1}W^{(h)}$
+
$h_t = \frac{e^{(w^{(p)}p_t + h_{t-1}w^{(h)}+b^{(h)})} - e^{-(w^{(p)}p_t + h_{t-1}w^{(h)}+b^{(h)})}}{e^{(w^{(p)}p_t + h_{t-1}w^{(h)}+b^{(h)})} + e^{-(w^{(p)}p_t + h_{t-1}w^{(h)}+b^{(h)})}}$
h_t
$W^{(p)} p_t$
p_t

Figure 2.
A typical RNN that has a hyperbolic tangent activation function $\left(\frac{e^{(x)} - e^{-(x)}}{e^{(x)} + e^{-(x)}}\right)$ *to generate the hidden state. Because of the hidden state RNNs have a "memory" that information has been calculated so far is captured. The information in hidden state passed further to a second activation function* $\left(\frac{1}{1+e^{-(x)}}\right)$ *to generate the predicted (output) values. In RNNs, the weight* (**W**) *calculation of each time point of the network model is related to the content of the previous time point. We can process a sequence of vectors of inputs* (**p**) *by applying a recurrence formula at every time step.*

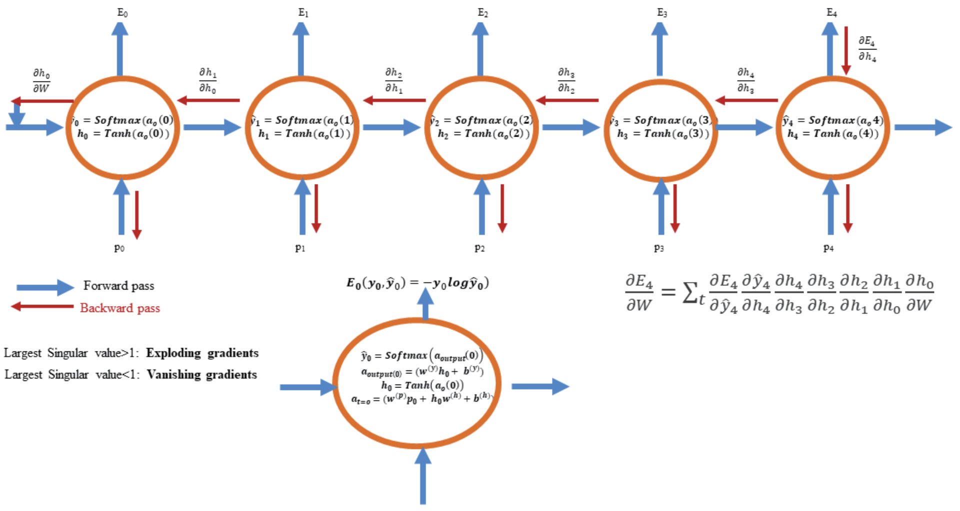

E₀
E₁
E₂
E₃
E₄
$\frac{\partial h_0}{\partial W}$
$\frac{\partial h_1}{\partial h_0}$
$\frac{\partial h_2}{\partial h_1}$
$\frac{\partial h_3}{\partial h_2}$
$\frac{\partial h_4}{\partial h_3}$
$\frac{\partial E_4}{\partial h_4}$
$\hat{y}_0 = Softmax(a_o(0))$ $h_0 = Tanh(a_o(0))$
$\hat{y}_1 = Softmax(a_o(1))$ $h_1 = Tanh(a_o(1))$
$\hat{y}_2 = Softmax(a_o(2))$ $h_2 = Tanh(a_o(2))$
$\hat{y}_3 = Softmax(a_o(3))$ $h_3 = Tanh(a_o(3))$
$\hat{y}_4 = Softmax(a_o(4))$ $h_4 = Tanh(a_o(4))$
p₀
p₁
p₂
p₃
p₄
Forward pass
Backward pass
$E_0(y_0,\hat{y}_0) = -y_0 log\hat{y}_0)$
$\frac{\partial E_4}{\partial W} = \sum_t \frac{\partial E_4}{\partial \hat{y}_4}\frac{\partial \hat{y}_4}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial h_0}\frac{\partial h_0}{\partial W}$
Largest Singular value>1: Exploding gradients
Largest Singular value<1: Vanishing gradients
$\hat{y}_0 = Softmax(a_{output}(0))$
$a_{output(0)} = (w^{(y)}h_0 + b^{(y)})$
$h_0 = Tanh(a_o(0))$
$a_{t=0} = (w^{(p)}p_0 + h_0 w^{(h)} + b^{(h)})$

Figure 3.
An unrolled RNN with a hidden state carries pertinent information from one input item in the series to others. The blue and red arrows in the figure are indicating the forward and the backward pass of the network, respectively. With backward pass, we sum up the contributions of each time step to the gradient. In other words, because W is used in every step up to the output, we need to backpropagate gradients from t = 4 *through the network all the way to* t = 0.

The weight of the previous hidden state and current input are placed in a trainable weight matrix. Element-wise multiplication of the previous hidden state vector with the hidden state weights $(\boldsymbol{h_{t-1}W^{(h)}})$ and element wise multiplication of the current input vector with the current input weights $(\boldsymbol{W^{(p)}p_t})$ produces the parameterized of state vector and input vector.

2. Hyperbolic tangent activation function is applied to the summed of the two parameterized vectors $\left(\boldsymbol{W^{(p)}p_t} + \boldsymbol{h_{t-1}W^{(h)}} + \boldsymbol{b^{(h)}}\right)$ to push the output between −1 and 1 (**Figure 2**).

$$f(a_h(t)) = h_t = \frac{e^{\left(W^{(p)}p_t+h_{t-1}W^{(h)}+b^{(h)}\right)} - e^{-\left(W^{(p)}p_t+h_{t-1}W+b^{(h)}\right)}}{e^{\left(W^{(p)}p_t+h_{t-1}W^{(h)}+b^{(h)}\right)} + e^{-\left(W^{(p)}p_t+h_{t-1}W+b^{(h)}\right)}} \tag{2}$$

3. The network input to the output unit at time t with element-wise multiplication of output weights and with updated (current) hidden state $\left(\boldsymbol{h_t W^{(y)}}\right)$.

   Therefore, the value before a softmax activation function takes place is $\boldsymbol{a_o(t)} = \boldsymbol{h_t W^{(y)}} + \boldsymbol{b^{(y)}}$. Here $\boldsymbol{W^{(y)}}$ and $\boldsymbol{b^{(y)}}$ are the weight and bias of the output layer.

4. The output of the network at time t is calculated (the activation function applied to the output layer depends on the type of target (dependent) variable and the values coming from the hidden units. Again, a second activation function (mostly sigmoid) is applied to the value generated by the hidden node. The predicted value of a RNN block with sigmoid:

$$\boldsymbol{\hat{y}_t} = \frac{\boldsymbol{1}}{\boldsymbol{1} + \boldsymbol{e}^{\left(\boldsymbol{W^{(y)}h_t} + \boldsymbol{b^{(y)}}\right)}} \tag{3}$$

During the training of the forward pass of the RNN, the network outputs predicted value $(\hat{y}_i, i = t-1, t, t+1, \dots .., t+s)$ at each time step. We can image the unfold (unroll) of RNN given in **Figure 3**. That is, for each time step, an RNN can be imaged as multiple copies of the same network for the complete sequence. For example, if the sequence is a sentence of four words as; "*I have kidney problem*" then the RNN would be unrolled into a 4-layer neural network, one layer for each word. The output given in (3) is used to train the network using gradient descent after calculation of error in (4).

5. Then the error (loss function, cost function) at each time step is calculated to start the "*backward pass*":

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t) \tag{4}$$

Here y and $\hat{y}_t$ are actual and predicted outcomes, respectively. After calculation of the error at each time step, this calculated error is injected backwards into the network to update the network weights at each epoch (iteration). As there are many training algorithms based on some modification of standard backpropagation, the chosen error measure can be different and depends on the selected algorithm. For example, the error $\boldsymbol{E_t}(\mathrm{y}) = \boldsymbol{E_t}(\boldsymbol{y_t}, \boldsymbol{\hat{y}_t})$ given in (4), has an additional term, $E_t(w)$, in the Bayesian regularized neural networks (BRANN) training algorithm that penalizes large weights in anticipation of achieving smoother mapping. Both $E_t(y)$, $E_t(w)$, have a coefficient as β and α, respectively (also referred to as regularization parameters or hyper-parameters) that need to be estimated adaptively [1, 16].

Stage 2: *Backward Pass.*

After the forward pass of RNN, the calculated error (loss function, cost function) at each time step is injected backwards into the network to update the network weights at each iteration. The idea of RNN unfolding in **Figure 3** takes place the bigger part in the way RNNs are implemented for the backward pass. Like standard backpropagation in feed forward MLP, the backward pass consists of a repeated application of the chain rule. For this reason, the type of backpropagation algorithm used for an RNN to update the network parameters is called backpropagation through time (BPTT). In BPTT, the RNN network is unfolded in time to construct a feed forward MLP neural network. Then, the generalized delta rule is applied to update the weights $\boldsymbol{W^{(p)}}$, $\boldsymbol{W^{(h)}}$ and $\boldsymbol{W^{(y)}}$ and biases $\boldsymbol{b^{(h)}}$ and $\boldsymbol{b^{(y)}}$. Remember, the goal with backpropagation is minimizing the gradients of error with respect to the network parameter space ($\boldsymbol{W^{(p)}}$, $\boldsymbol{W^{(h)}}$ and $\boldsymbol{W^{(y)}}$ and biases $\boldsymbol{b^{(h)}}$ and $\boldsymbol{b^{(y)}}$) and then updates the parameters using Stochastic Gradient Descent. The following equation is used to update the parameters for minimizing the error function:

$$W_t = W_{t-1} - a\frac{\partial E}{\partial W}.$$

Here, a is learning rate and $\frac{\partial E}{\partial W}$ is the derivative of the error function with respect to parameters space. The same is applied to all weights and biases on the networks. The error each time step is

$$E_0(y, \hat{y}) = -ylog\hat{y}$$

The total error is calculated by the summation of the error from all time steps as:

$$E = \sum_t - y_t \, log \, \hat{y}_t \tag{5}$$

The value of gradients $\frac{\partial E}{\partial W}$ at each time step is calculated as (the same rule is applied to all parameters on the network):

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W} \tag{6}$$

To calculate the error gradient given in Eq. (6):

$$\cdot \qquad \frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial h_{t-2}}\frac{\partial h_{t-2}}{\partial h_{t-3}} \dots \dots \dots \dots . \frac{\partial h_0}{\partial W}$$

To calculate the overall error gradient, the chain rule of differentiation given in (7) is used.

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial h_{t-2}}\frac{\partial h_{t-2}}{\partial h_{t-3}} \dots \dots \dots \dots . \frac{\partial h_0}{\partial W} \tag{7}$$

Then the network weights can be updated as follow:

$$W_{t+1}^{(h)} = W_t^{(h)} + a\frac{\partial E}{\partial W^{(h)}}$$

$$W_{t+1}^{(p)} = W_t^{(p)} + a\frac{\partial E}{\partial W^{(p)}}$$

$$W_{t+1}^{(y)} = W_t^{(y)} + a\frac{\partial E}{\partial W^{(y)}}$$

Note that, as given in (2), the current state (h_t) = $tanh\left(W^{(p)}p_t + h_{t-1}W^{(h)} + b^{(h)}\right)$

depends on the quantity of the previous state $(h_{t\text{-}1})$ and the other parameters. Therefore, the differentiation of h_t and h_j (here j = 0, 1, ..., t-1) given in (7) is a derivative of a hidden state that stores memory at time t.

The Jacobians of any time $\left(\frac{\partial h_j}{\partial h_{j-1}}\right)$ and for the entire time will be:

$$\frac{\partial h_j}{\partial h_{j-1}} = \frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial h_{t-2}}\frac{\partial h_{t-2}}{\partial h_{t-3}} \dots \dots . \frac{\partial h_{j+1}}{\partial h_j} = \prod_{j=j+1}^{t}\frac{\partial h_j}{\partial h_{j-1}} \tag{8}$$

while the Jacobian matrix for hidden state is given by:

$$\frac{\partial h_j}{\partial h_{j-1}} = \left[\frac{\partial h_j}{\partial h_{j-1,1}}, \frac{\partial h_j}{\partial h_{j-1,2}} \cdots \frac{\partial h_j}{\partial h_{j-1,s}}\right] = \begin{bmatrix} \frac{\partial h_{j,1}}{\partial h_{j-1,1}} & \cdots & \frac{\partial h_{j,1}}{\partial h_{j-1,s}} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_{j,s}}{\partial h_{j-1,1}} & \cdots & \frac{\partial h_{j,s}}{\partial h_{j-1,s}} \end{bmatrix} \tag{9}$$

Putting the Eqs. (7) and (8) together, we have the following relationship:

$$\frac{\partial E}{\partial W} = \sum_t \sum_j \frac{\partial E_t}{\partial y_t}\frac{\partial y_t}{\partial h_t}\left(\prod_{j=j+1}^{t}\frac{\partial h_j}{\partial h_{j-1}}\right)\frac{\partial h_j}{\partial W}. \tag{10}$$

In other words, because the network parameters are used in every step up to the output, we need to backpropagate gradients from last time step (t = t) through the network all the way to t = 0. The Jacobians in (10), $\left(\frac{\partial h_j}{\partial h_{j-1}}\right)$, demonstrates the eigen decomposition given by $\boldsymbol{W^{(i)T}diag}\left(\boldsymbol{f'}\left(\boldsymbol{h_{j-1}}\right)\right)$, where the eigenvalues and eigenvectors are generated. Here the $W^{(i)T}$ is the transpose of the network parameters matrix. Consequently, if the largest eigenvalue is greater or smaller than 1, the RNN suffers from vanishing or exploding gradient problems (see **Figure 3**).

4. Long short-term memory

As mentioned before, the output from RNNs is dependent on its previous state or previous N time steps circumstances. Conventional RNN face difficulty in learning and maintaining long-range dependencies. Imagine the unfolding RNN given in **Figure 3**. Each time step requires a new copy of the network. With large RNNs, thousands, even millions of weights are needed to be updated. In other word, $\frac{\partial h_j}{\partial h_{j-1}}$ is a chain rule itself. For **Figure 3**, for example, the derivative of $\frac{\partial h_4}{\partial h_3} = \frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial h_0}$. Imagine an unrolling the RNN a thousand times, in which every activation of the neurons inside the network are replicated thousands of times. This means, especially for larger networks, that thousands or millions of weights are needed. As Jacobian matrix will play a role to update the weights, the values of the Jacobian matrix will range between −1, 1 if tanh activation function is applied to the $f(a_h(t)) = h_t = tanh\left(\boldsymbol{W}^{(y)}h_t + b^{(y)}\right)$ given in (2). It can be easily imagined that the derivatives of tanh (or sigmoid) activation function would be 0 at the end. Zero gradients drive other gradients in previous layers towards 0. Thus, with small values in the Jacobian matrix and multiple matrix multiplications (t-j, in particular) the gradient values will be shrunk exponentially fast, eventually vanishing completely after a few time steps. As a result, the RNN ends up not learning long-range dependencies. As in RNNs, the vanishing gradients problem will be an important issue for the deep feedforward MLP when multiple hidden layers (multiple neurons within each) are placed between input and output layers.

The long short-term memory networks (LSTMs) are a special type of RNN that can overcome the vanishing gradient problem and can learn long-term dependencies. LSTM introduces a memory unit and a gate mechanism to enable capture of the long dependencies in a sequence. The term "long short-term memory" originates from the following intuition. Simple RNN networks have long-term memory in the form of weights. The weights change gradually during the training of the network, encoding general knowledge about the training data. They also have short-term memory in the form of ephemeral activations, which flows from each node to successive nodes [17, 18].

4.1 The architecture of LSTM

The neural network architecture for an LSTM block given in **Figure 4** demonstrates that the LSTM network extends RNN's memory and can selectively remember or forget information by structures called cell states and three gates. Thus, in addition to a hidden state in RNN, an LSTM block typically has four more layers. These layers are called the cell state ($\boldsymbol{C_t}$), an input gate ($\boldsymbol{i_t}$), an output gate ($\boldsymbol{O_t}$), and a forget gate ($\boldsymbol{f_t}$). Each layer interacts with each other in a very special way to generate information from the training data.

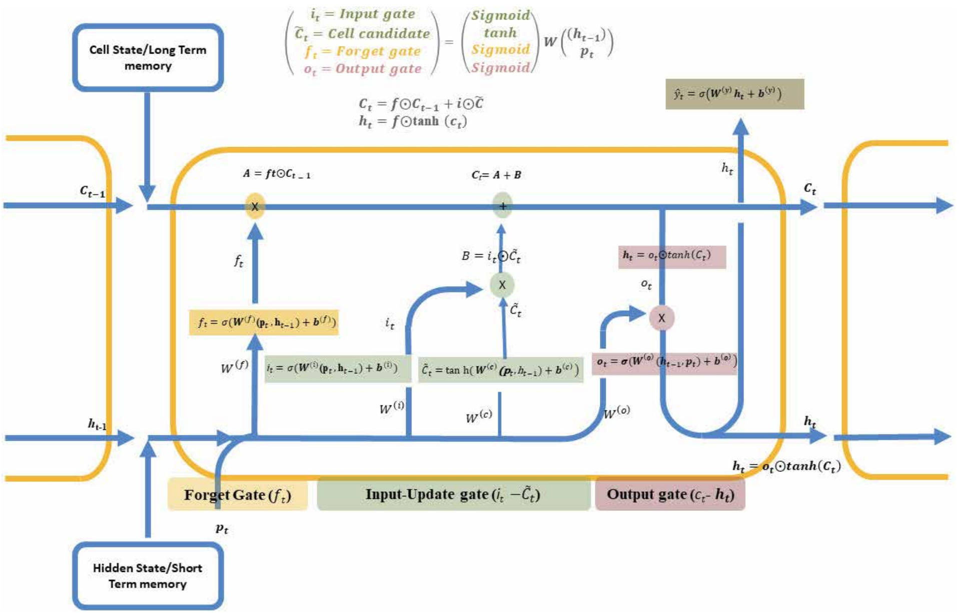

i_t = Input gate
$\tilde{C}_t$ = Cell candidate
f_t = Forget gate
o_t = Output gate
Sigmoid
tanh
Sigmoid
Sigmoid
$W\begin{pmatrix} h_{t-1} \\ p_t \end{pmatrix}$
Cell State/Long Term memory
$C_t = f \odot C_{t-1} + i \odot \tilde{C}$
$h_t = f \odot \tanh(c_t)$
$\hat{y}_t = \sigma(W^{(y)} h_t + b^{(y)})$
C_{t-1}
$A = ft \odot C_{t-1}$
$C_t = A + B$
h_t
C_t
f_t
$B = i_t \odot \tilde{C}_t$
$h_t = o_t \odot tanh(C_t)$
o_t
$\tilde{C}_t$
$f_t = \sigma(W^{(f)}(p_t, h_{t-1}) + b^{(f)})$
i_t
$W^{(f)}$
$i_t = \sigma(W^{(i)}(p_t, h_{t-1}) + b^{(i)})$
$\tilde{C}_t = \tanh(W^{(c)}(p_t, h_{t-1}) + b^{(c)})$
$o_t = \sigma(W^{(o)}(h_{t-1}, p_t) + b^{(o)})$
$W^{(i)}$
$W^{(c)}$
$W^{(o)}$
h_{t-1}
h_t
$h_t = o_t \odot tanh(C_t)$
Forget Gate (f_t)
Input-Update gate ($i_t - \tilde{C}_t$)
Output gate (c_t- h_t)
p_t
Hidden State/Short Term memory

Figure 4.
Illustration of long short-term memory block structure. The operator "$\odot$" denotes the element-wise multiplication. The C_{t-1}, C_t, h_t and h_t are previous cell state, current cell state, current hidden state and previous hidden state, respectively. The f_t; i_t; o_t are the values of the forget, input and output gates, respectively. The $\tilde{C}_t$ is the candidate value for the cell state, $W^{(f)}$, $W^{(i)}$, $W^{(c)}$, $W^{(o)}$ are weight matrices consist of forget gate, input gate, cell state and output gate weights, and $b^{(f)}$, $b^{(i)}$, $b^{(c)}$, and $b^{(o)}$ are bias vectors associated with them.

A block diagram of LSTM at any timestamp is depicted in **Figure 4**. This block is a recurrently connected subnet that contains the same cell state and three gates structure. The $\boldsymbol{p}_t$, $\boldsymbol{h}_{t-1}$, and $\boldsymbol{C}_{t-1}$ correspond to the input of the current time step, the hidden output from the previous LSTM unit, and the cell state (memory) of the previous unit, respectively. The information from the previous LSTM unit is combined with current input to generate a newly predicted value. The LSTM block is mainly divided into three gates: forget (blue), input-update (green), and output (red). Each of these gates is connected to the cell state to provide the necessary information that flows from the current time step to the next.

A sigmoid activation function $\left(\frac{1}{1+e^{-(x)}}\right)$ is implemented in the forget gate. For the input and output gates, however, a combination of sigmoid and hyperbolic tangent-tanh-$\left(\frac{e^{(x)}-e^{-(x)}}{e^{(x)}+e^{-(x)}}\right)$ are used to provide the necessary information to the cell state. The information generated by the blocks flow through the cell state from one block to another as the chain of repeating components of the LSTM neural network holds. Details about cell state and each layer are given in different subtitles.

4.1.1 Cell state

As shown in the upper part of **Figures 4** and **5**, the cell state is the key to LSTMs and represents the memory of LSTM networks. The process for the cell state is very much like to a conveyor belt or production chain. The information about the parameters runs straight forward the entire chain, with only some linear interactions, such as multiplication and addition. The state of information depends on these interactions. If there are no interactions, the information will run along without changes. The LSTM block removes or adds information to the cell state through the gates, which allow optional information to cross [19].

4.1.2 Forget gate

The *Forget Gate* ($\boldsymbol{f}_t$) decides the type of information that should be thrown away or kept from the cell state. This process is implemented by a sigmoid activation function. The sigmoid activation function outputs values between 0 and 1 coming from the weighted input ($\mathbf{W}_f \boldsymbol{p}_t$), previous hidden state ($\boldsymbol{h}_{t-1}$), and a bias ($\boldsymbol{b}_f$). The forget gates (**Figure 6**) can be described by the equation given in (11). Here, σ is the

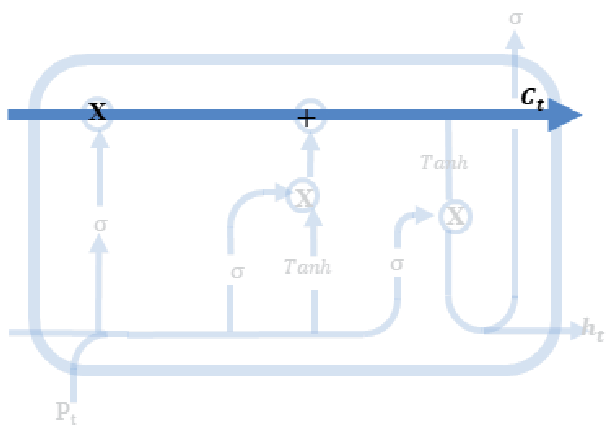


Figure 5.
The cell state, the horizontal line running through the top of the diagram of an LSTM.

sigmoid activation function, $\boldsymbol{W}^{(f)}$ and $\boldsymbol{b}^{(f)}$ are the weight matrix and bias vector, which will be learned from the input training data.

$$f_t = \sigma\left(W^{(f)}(\mathrm{p}_t, \mathrm{h}_{t-1}) + b^{(f)}\right) = \frac{1}{1 + e^{-\left(W^{(f)}.(\mathrm{p}_t,\mathrm{h}_{t-1})+b^{(f)}\right)}} \tag{11}$$

The function takes the old output ($\boldsymbol{h}_{t-1}$) at time $t - 1$ and the current input ($\boldsymbol{p}_t$) at time t for calculating the components that control the cell state and hidden state of the layer. The results are [0,1], where 1 represents "completely hold this" and 0 represents "completely throw this away" (**Figure 6**).

4.1.3 Input gate

The *Input Gate* (i_t) controls what new information will be added to the cell state from the current input. This gate also plays the role to protect the memory contents from perturbation by irrelevant input (**Figures** 7 and 8). A sigmoid activation function is used to generate the input values and converts information between 0 and 1. So, mathematically the input gate is:

$$i_t = \sigma\left(W^{(i)}(p_t, h_{t-1}) + b^{(i)}\right) = \frac{1}{1 + e^{-\left(W^{(i)}.(p_t,h_{t-1})+b^{(i)}\right)}} \tag{12}$$

where $\boldsymbol{W}^{(\mathrm{i})}$ and $\boldsymbol{b}^{(\mathrm{i})}$ are the weight matrix and bias vector, $\boldsymbol{p}_t$ is the current input timestep index with the previous time step $\boldsymbol{h}_{t\text{-}1}$. Similar to the forget gate, the parameters in the input gate will be learned from the input training data. At each time step, with the new information $\boldsymbol{p}_t$, we can compute a candidate cell state.

Next, a vector of new candidate values, $\tilde{C}_t$, is created. The computation of the new candidate is similar to that of (11) and (12) but uses a hyperbolic tanh activation function with a value range of (−1,1). This leads to the following Eq. (13) at time t.

$$\tilde{C}_t = tanh\left(W^{(c)}(p_t, h_{t-1}) + b^{(c)}\right) = \frac{e^{\left(W^{(c)}.(p_t,h_{t-1})+b^{(c)}\right)} - e^{-\left(W^{(c)}.(p_t,h_{t-1})+b^{(c)}\right)}}{e^{\left(W^{(c)}.(p_t,h_{t-1})+b^{(c)}\right)} + e^{-\left(W^{(c)}.(p_t,h_{t-1})+b^{(c)}\right)}} \tag{13}$$

In the next step, the values of the input state and cell candidate are combined to create and update the cell state as given in (14). The linear combination of the input gate and forget gate are used for updating the previous cell state ($\boldsymbol{C}_{t\text{-}1}$) into current

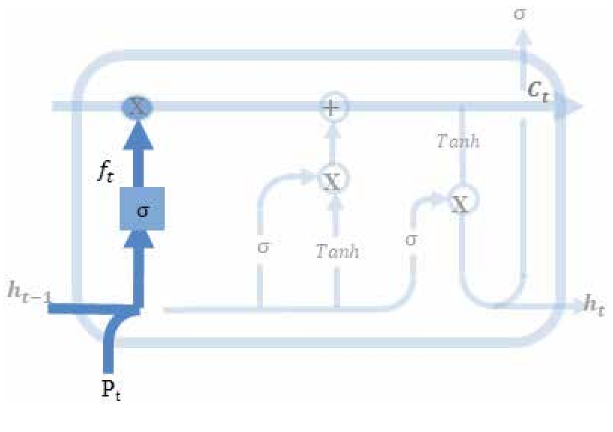


Figure 6.
The forget gate controls what information to throw away from the memory.

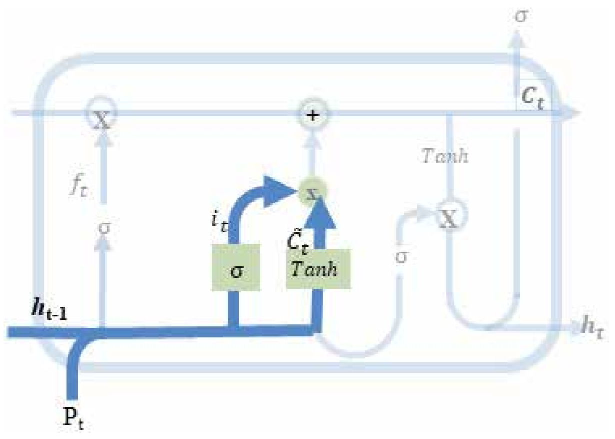


Figure 7.
The input-update gate decides what new information should be stored in the cell state, which has two parts: A sigmoid layer and a hyperbolic tangent (tanh) layer. The sigmoid layer is called the "input gate layer" because it decides which values should be updated. The tanh layer is a vector of new candidate values $\tilde{C}_t$ that could be added to the cell state.

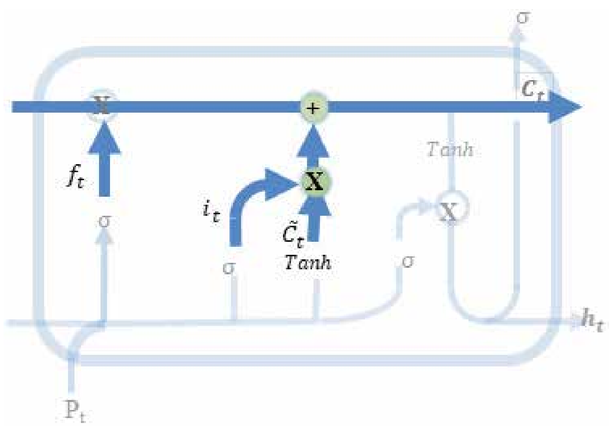


Figure 8.
Memory update is done using old memory via the forget gate and new memory via the input gate.

cell state ($\boldsymbol{C_t}$). Once again, the input gate ($\boldsymbol{i_t}$) governs how much new data should be taken into account via the candidate ($\tilde{C}_t$), while the forget gate ($\boldsymbol{f_t}$) reports how much of the old memory cell content ($\boldsymbol{C}_{t\text{-}1}$) should be retained. Using the same pointwise multiplication ($\odot$ =Hadamard product), we arrive at the following updated equation:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{14}$$

4.1.4 Output gate

The *Output Gate* ($\boldsymbol{o_t}$) controls which information to reveal from the updated cell state ($\boldsymbol{C_t}$) to the output in a single time step. In other words, the output gate determines what the value of the next hidden state should be in each time step. As depicted in **Figure 9**, the hidden state comprises information on previous inputs. Moreover, the calculated value of the hidden state for the given time step is used for

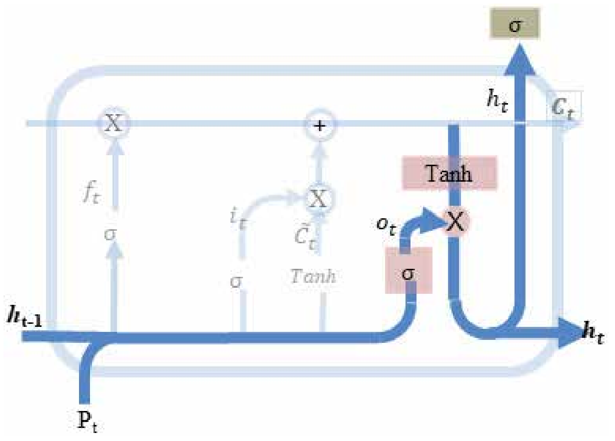


Figure 9.
The output state decides what information will be output using a sigmoid σ and tanh (to push the values to be between −1 and 1) layers.

the prediction ($\hat{y}_t = \text{softmax}(.)$). Here, softmax is a nonlinear activation function (sigmoid, hyperbolic tangent etc.).

$$o_t = \sigma\left(W^{(o)}(h_{t-1}, p_t) + b^{(o)}\right) = \frac{1}{1 + e^{-\left(W^{(o)}(h_{t-1}, p_t) + b^{(o)}\right)}} \tag{15}$$

$$h_t = o_t \odot tanh\,(C_t) = o_t \bullet \frac{e^{(f_t \odot C_{t-1} + i_t \odot \tilde{C}_{t})} - e^{-(f_t \odot C_{t-1} + i_t \odot \tilde{C}_{t})}}{e^{(f_t \odot C_{t-1} + i_t \odot \tilde{C}_{t})} + e^{-(f_t \odot C_{t-1} + i_t \odot \tilde{C}_{t})}} \tag{16}$$

$$\hat{y}_t = \sigma\left(W^{(y)} h_t + b^{(y)}\right) = \frac{1}{1 + e^{-\left(W^{(y)} h_t + b^{(y)}\right)}} \tag{17}$$

First, the previous hidden state ($\boldsymbol{h}_{t\text{-}1}$) is passed to the current input into a sigmoid function. Next newly updated cell state is generated with the tanh function [15, 18]. Finally, the tanh output is multiplied with the sigmoid output to determine what information the hidden state should carry (16). The final product of the output gate is an updated of the hidden state, and this is used for the prediction at time step t. Therefore, the aim of this gate is to separate the updated cell state (updated memory) from the hidden state. The updated cell state ($\boldsymbol{C}_t$) contains a lot of information that is not necessarily required to be saved in the updated hidden state. However, this information is critical as the updated hidden state at each time is used in all gates of an LSTM block. Thus, the output gate does the assessment regarding what parts of the cell state ($\mathbf{C}_t$) is presented in the hidden state ($\boldsymbol{h}_t$). The new cell and new hidden states are then passed to the next time step (**Figure 9**).

Summary of forward pass

1. *Forget gate*: Controls what information to throw away and decides how much from the part should be remember. $\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}^{(f)}(\mathbf{p}_t, \mathbf{h}_{t-1}) + \boldsymbol{b}^{(f)}\right)$

2. *Input-Update Gate:* Controls information to add cell state from current input and decides how much should be added to the cell state $\boldsymbol{\imath}_t = \sigma\left(\boldsymbol{W}^{(i)}(\mathbf{p}_t, \mathbf{h}_{t-1}) + \boldsymbol{b}^{(i)}\right), \tilde{C}_t = \tanh\left(\boldsymbol{W}^{(c)}(\mathbf{p}_t, \mathbf{h}_{t-1}) + \boldsymbol{b}^{(c)}\right)$

3. *Output gate:* Determines the part of the current cell state makes it *to the output* $\boldsymbol{o_t} = \boldsymbol{\sigma}\left(\boldsymbol{W^{(o)}}(\boldsymbol{h_{t-1}}, \boldsymbol{p_t}) + \boldsymbol{b^{(o)}}\right).$

4. *Current cell state:* $\boldsymbol{C_t} = \boldsymbol{f_t} \odot \boldsymbol{C_{t-1}} + \boldsymbol{i_t} \odot \tilde{\boldsymbol{C}}_t$

5. *Current hidden state:* $\boldsymbol{h_t} = \boldsymbol{o_t} \odot \boldsymbol{tanh}\,(\boldsymbol{C_t}) \Rightarrow \boldsymbol{h_t} = \boldsymbol{LSTM}((\boldsymbol{p_t}, \boldsymbol{h_{t-1}})$

6. *LSTM block prediction:* $\hat{y}_t = \sigma\left(\boldsymbol{W^{(y)}}\boldsymbol{h_t} + \boldsymbol{b^{(y)}}\right)$

7. *Calculate the LSTM block error for the time step:* $\boldsymbol{E_t}(\boldsymbol{y_t}, \hat{\boldsymbol{y}}_t) = -\boldsymbol{y_t}\,\boldsymbol{log}\,\hat{\boldsymbol{y}}_t)$

4.2 Backward pass

Like the RNN networks, an LSTM network generates an output $(\hat{y}_t)$ at each time step that is used to train the network via gradient descent (**Figure 10**). During the backward pass, the network parameters are updated at each epoch (iteration). The only fundamental difference between the back-propagation algorithms of the RNN and LSTM networks is a minor modification of the algorithm. Here, the calculated error term at each time step is $\boldsymbol{E_t} = -\boldsymbol{y_t}\,\boldsymbol{log}\,\hat{\boldsymbol{y}}_t$ **.** As in RNN, the total error is calculated by the summation of error from all time steps $\boldsymbol{E} = \sum_t - \boldsymbol{y_t}\,log\,\hat{\boldsymbol{y}}_{t.}$

Similarly, the value of gradients $\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{W}}$ at each time step is calculated and then the summation of the gradients at each time steps $\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{W}} = \sum_t \frac{\partial \boldsymbol{E_t}}{\partial \boldsymbol{W}}$ is obtained. Remember, the predicted value, $\hat{y}_t$, is a function of the hidden state $(\hat{y}_t = \sigma\left(W^{(y)}\boldsymbol{h_t} + b^{(y)}\right))$ and the hidden state (h_t) is a function of the cell state $(h_t = o_t \odot tanh\,(\boldsymbol{C_t}))$.These both are subjected in the chain rule. Hence, the derivatives of individual error terms with respect the network parameter:

$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\frac{\partial h_t}{\partial c_t}\frac{\partial c_t}{\partial c_{t-1}}\frac{\partial c_{t-1}}{\partial c_{t-2}}\frac{\partial c_{t-2}}{\partial c_{t-3}} \dots\dots\dots\dots \frac{\partial c_0}{\partial W} \tag{18}$$

and for the overall error gradient using the chain rule of differentiation is:

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\frac{\partial h_t}{\partial c_t}\frac{\partial c_t}{\partial c_{t-1}}\frac{\partial c_{t-1}}{\partial c_{t-2}}\frac{\partial c_{t-2}}{\partial c_{t-3}} \dots\dots\dots\dots \frac{\partial c_0}{\partial W} \tag{19}$$

As Eq. (19) illustrates, the gradient involves the chain rule of $\partial \boldsymbol{c_t}$ in an LSTM training using the backpropagation algorithm, while the gradient equation involves a chain rule of $\partial \boldsymbol{h_t}$ for a basic RNN. Therefore, the Jacobian matrix for cell state for an LSTM is [20]:

$$\left[\frac{\partial c_j}{\partial c_{j-1,1}}, \frac{\partial c_j}{\partial c_{j-1,2}} \cdots \frac{\partial c_j}{\partial c_{j-1,s}}\right] = \begin{bmatrix} \frac{\partial c_{j,1}}{\partial c_{j-1,1}} & \cdots & \frac{\partial c_{j,1}}{\partial c_{j-1,s}} \\ \vdots & \ddots & \vdots \\ \frac{\partial c_{j,s}}{\partial c_{j-1,1}} & \cdots & \frac{\partial c_{j,s}}{\partial c_{j-1,s}} \end{bmatrix} \tag{20}$$

The problem of gradient vanishing

Recall the Eq. (14) for cell state is $\boldsymbol{C_t} = \boldsymbol{f_t} \odot \boldsymbol{C_{t-1}} + \boldsymbol{i_t} \odot \tilde{\boldsymbol{C}}_t$. When we consider Eq. (19), the value of the gradients in the LSTM is controlled by the chain of

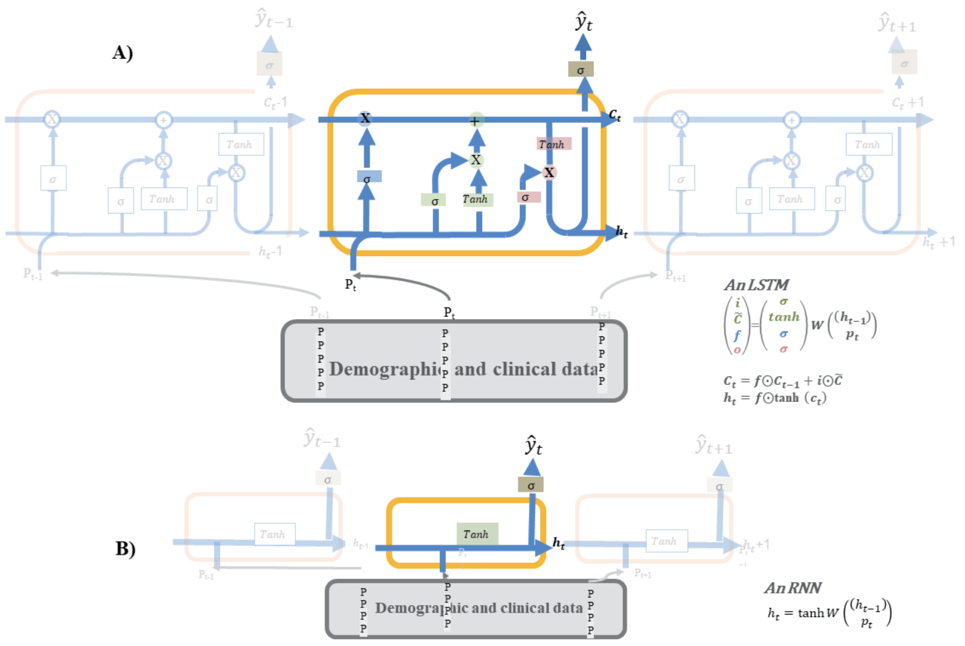

A)
B)
Demographic and clinical data
An LSTM
$C_t = f \odot C_{t-1} + i \odot \tilde{C}$
$h_t = f \odot \tanh(c_t)$
An RNN
$h_t = \tanh W \binom{h_{t-1}}{p_t}$

Figure 10.
Illustration of the (A) an LSTM unit from 3-time steps with input data (demographic and clinical data). LSTM network takes inputs from to the current time step to update the hidden state and $(\mathbf{LSTM}((p_t, h_{t-1}))$ *with relevant information. The "x" in the circles denote point-wise operators, σ and tanh are sigmoid* $\left(\left(\frac{1}{1+e^{-(x)}}\right)\right.$, *generates between 0 and 1) and hyperbolic tangent* $\left(\left(\frac{e^{(x)}-e^{-(x)}}{e^{(x)}+e^{-(x)}}\right)\right.$,*generates between* -1 *and 1) activation functions. (B) an RNN with 3-time steps. It has only a tangent,* $\left(\frac{e^{(x)}-e^{-(x)}}{e^{(x)}+e^{-(x)}}\right)$, *activation function in the block.*

derivatives starting from the part $\frac{\partial c_t}{\partial c_{t-1}}$. Expanding this value using the expression for $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$.

$$\begin{aligned}\frac{\partial c_t}{\partial c_{t-1}} &= \frac{\partial(f_t \odot C_{t-1} + i_t \odot \tilde{C}_t)}{\partial(f_{t-1} \odot C_{t-2} + i_{t-1} \odot \tilde{C}_{t-1})} \\ &= \frac{\partial c_t}{\partial f}\frac{\partial f}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial c_{t-1}} + \frac{\partial c_t}{\partial i}\frac{\partial i}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial c_{t-1}} + \frac{\partial c_t}{\partial \tilde{C}_t}\frac{\partial \tilde{C}_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial c_{t-1}} + \frac{\partial c_t}{\partial c_{t-1}}\end{aligned} \tag{21}$$

Note the term $\frac{\partial c_t}{\partial c_{t-1}}$ does not have a fixed pattern and can yield any positive value in the LSTM, while the $\frac{\partial c_t}{\partial c_{t-1}}$ term in the standard RNN can yield values greater than 1 or less than 1 after certain time steps. Thus, for an LSTM, the term will not converge to 0 or diverge completely, even for an infinite number of time steps. If the gradient starts converging towards zero, the weights of the gates are adjusted to bring it closer to 1.

4.3 Other type of LSTMs

Several modifications to original LSTM architecture have been recommended over the years. Surprisingly, the original continues to outperform, and has similar predictive ability compared with variants of LSTM over 20 years.

4.3.1 Peephole connections

This is a type of LSTM by adding "*peephole connections*" to the standard LSTM network. The theme stands for peephole connections needed to capture information inherent to time lags. In other words, with *peephole connections* the information conveyed by time intervals between sub-patterns of sequences is included to the network recurrent. Thus, *peephole connections* concatenate the previous cell state ($C_{t\text{-}1}$) information to the forget, input and output gates. That is, the expression of these gates with peephole connection would be:

$$\begin{aligned} \mathrm{f}_t &= \sigma\left(W^{(f)}\left(\mathrm{p}_t, \mathrm{h}_{t-1}, C_{t-1}\right) + b^{(f)}\right) \\ i_t &= \sigma\left(W^{(i)}\left(\mathrm{p}_t, \mathrm{h}_{t-1}, C_{t-1}\right) + b^{(i)}\right) \\ o_t &= \sigma\left(W^{(o)}\left(h_{t-1}, p_t\right) + b^{(o)}\right) \end{aligned} \tag{22}$$

This configuration was offered to improve the predictive ability of LSTMs to count and time distances between rare events [21].

4.3.2 Gated recurrent units

Gated recurrent units (GRU) is a simplified version of the standard LSTM designed in a manner to have more persistent memory in order to make it easier for RNNs. A GRU is called gated RNN and introduced by [22]. In a GRU, forget and input gates are merged into a single gate named "update gate". Moreover, the cell state and hidden state are also merged. Therefore, the GRU has fewer parameters and has been shown to outperform LSTM on some tasks to capture long-term dependencies.

4.3.3 Multiplicative LSTMs (mLSTMs)

This configuration of LSTM was introduced by Krause et al., [23]. The architecture is for sequence modeling combines LSTM and multiplicative RNN architectures. mLSTM is characterized by its ability to have different recurrent transition functions for each possible input, which makes it more expressive for autoregressive density estimation. Krause et al. concluded that mLSTM outperforms standard LSTM and its deep variants for a range of character-level language modeling tasks.

4.3.4 LSTM with attention

The core idea behind the LSTM with Attention frees the encoder-decoder architecture from the fixed-length internal illustration. This is one of the most transformative innovations in sequence to uncover the mobility regularities in the hidden node of LSTM. The LSTM with attention was introduced by Wu et al., [24] for Bridging the Gap between Human and Machine Translation in Google's Neural Machine Translation System. The LSTM with attraction consists of a deep LSTM network with 8 encoder and 8 decoder layers using attention and residual connections. Most likely, this type of LSTM continues to power Google Translate to this day.

4.4 Examples

Two examples using MATLAB for LSTM will be given for this particular chapter.

***Example* 1** This example shows how to forecast time series data for COVID19 in the USA using a long short-term memory (LSTM) network. The variable used in training data is the rate for the number of positive/number of tests for each day between 01/22/2020–2112/22/2020. Data set was taken from publicly available h ttps://covidtracking.com/data/national. web site and data are updated each day between about 6 pm and 7:30 pm Eastern Time Zone. The initiative relies upon publicly available data from multiple sources. States in the USA are not consistent in how and when they release and update their data, and some may even retroactively change the numbers they report. This can affect the predictions presented in these data visualizations (**Figure 11a-d**). The steps for example 1 are summarized in the **Table 1** (MATLAB 2020b) and results are illustrated in **Figure 11a-d.** LSTM network was trained on the first 90% of the sequence and tested on the last 10%. Therefore, results reveal predicting the positive last 38 days.

This example trains an LSTM network to forecast the number of positively tested given the number of cases in previous days. The training data contains a single time series, with time steps corresponding to days and values corresponding to the number of cases. To make predictions on a new sequence, reset the network state using the "*resetState*" command in MATLAB. Resetting the network state prevents previous predictions from affecting the predictions on the new data. Reset the network state, and then initialize the network state by predicting on the training data (MATLAB, 2020b). The solid line with red color in **Figure 11a** and **c** indicates the number of cases predicted for the last 30 days.

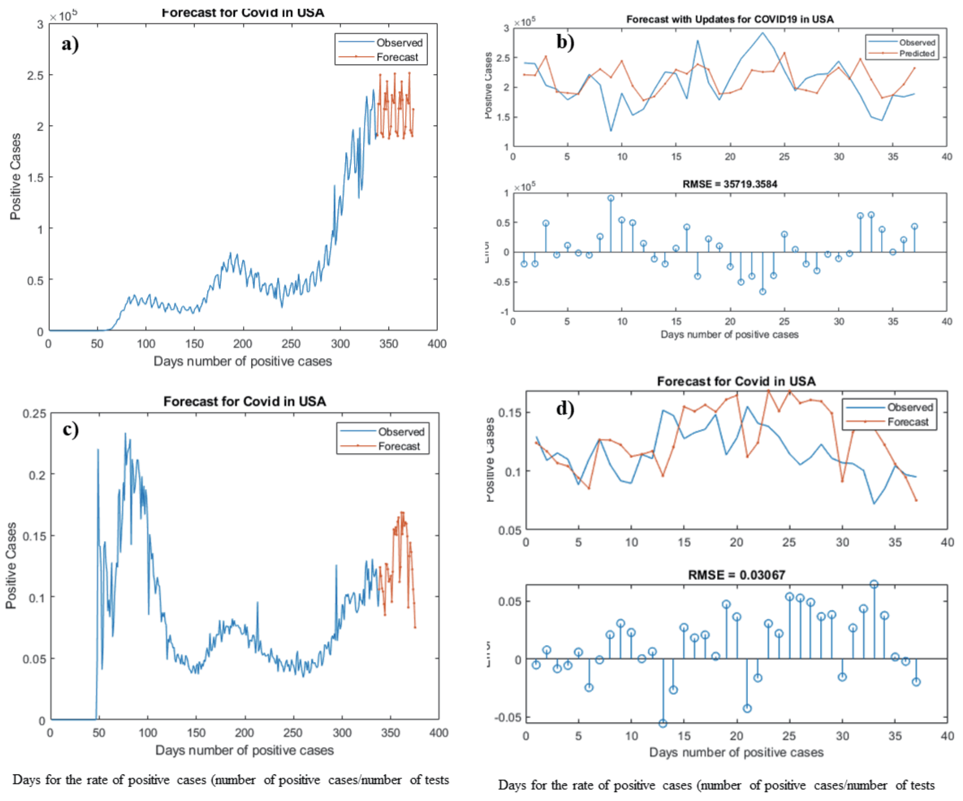


Figure 11.
Total daily number of positively tested COVID19 and the rate (positively tested/number of test) conducted in the USA. (a) Plot of the training time series of the number of positively tested COVID19 with the forecasted values, (b) compare the forecasted values of the number of positively tested with the test data set. This graph shows the total daily number of virus tests conducted in each state and of those tests, how many were positive each day. (c) Plot of the training time series of the rate of positively tested COVID19 (d) compare the forecasted values of the rate of positively tested with the rates in the test data set. The trend line in blue shows the actual number of positive cases and the trend line in red shows the number predicted for the last 38 days.

Example 1. MATLAB Codes and descriptions of the codes	
a. Data prepretion	**Descriptions**

numTimeStepsTrain = floor(0.95*numel(Tpositive));
dataTrain = Tpositive(1:numTimeStepsTrain+1);
dataTest = Tpositive(numTimeStepsTrain+1:end);
	Partition the training and test data. Train on the first 95% of the sequence and test on the last 5%

b. Define LSTM	**Descriptions**

numHiddenUnits = 300;
layers = [...sequenceInputLayer(numFeatures)
lstmLayer(numHiddenUnits)
fullyConnectedLayer(numResponses)
regressionLayer];
	Define LSTM Network Architecture and create an LSTM regression network. Specify the LSTM layer to have 300 hidden units

c. Specify the training options	**Descriptions**
options = *trainingOptions('adam'*, ... 'MaxEpochs',1000, ... 'GradientThreshold',1, ... 'InitialLearnRate',0.005, ... 'LearnRateSchedule','piecewise', ... 'LearnRateDropPeriod',125, ... 'LearnRateDropFactor',0.2, ... 'Verbose',0, ... 'Plots','training-progress');	Training with Adam (adaptive moment estimation) To prevent the gradients from exploding, set the gradient threshold to 1. The software updates the learning rate every certain number of epochs by multiplying with a certain factor. Number of epochs for dropping the learning rate Factor for dropping the learning rate when 'LearnRateSchedule','piecewise'
d. Train the LSTM network	**Descriptions**
net = trainNetwork(XTrain,YTrain,layers,options);	Train the LSTM network with the specified training options
e. Initialize -training and loop over	

net = predictAndUpdateState(net,XTrain);
[net,YPred] = predictAndUpdateState(net,YTrain (end));
numTimeStepsTest = numel(XTest);
for i = 2:numTimeStepsTest
[net,YPred(:,i)] = predictAndUpdateState(net,YPred (:,i-1),'ExecutionEnvironment','cpu'); end;
	The training and update data Next, make the first prediction using the last time step of the training response YTrain(end). Loop over the remaining predictions and input the previous prediction to predictAndUpdateState.

f. Update Network State with observed values	**Descriptions**

net = resetState(net);
net = predictAndUpdateState(net,XTrain);
	Update Network State with Observed Values

g. Predict on each time step	

YPred = [];
numTimeStepsTest = numel(XTest);
for i = 1:numTimeStepsTest
[net,YPred(:,i)] = predictAndUpdateState(net,XTest (:,i),'ExecutionEnvironment','cpu');
end
	Predict on each time step. For each prediction, predict the next time step using the observed value of the previous time step.

*All descriptions are based on MATLAB 2020b and related examples from the MATLAB.[*]	

Table 1.
MATLAB codes and specification of cods for Example 1[*].

***Example* 2**: Data from example 2 is from SEER 2017 for different age groups. SEER collects cancer incidence data from population-based cancer registries of the U.S. population. The SEER registries collect data on patient demographics, primary tumor site, tumor morphology, stage at diagnosis, and the first course of treatment, and they follow up with patients for vital status. The example given in this chapter is the cancer type and non-cancer causes of death identified from the survey. https://seer.cancer.gov/data/. The steps for example 2 are summarized in **Table 2** (MATLAB 2020b) and because of space limitation, only the cloud of cancer from

Example 2. MATLAB Codes and descriptions of the codes	
a. Data prepretion	**Descriptions**

filename = "filename"
data = readtable(filename,'TextType','string');
	To import the text data as strings, specify the text type to be 'string'.

b. Partition data	**Descriptions**

crossval = cvpartition('DataName.
ClassVaribleName',0.30);
dataTrain = DataName(training(crossval),:);
dataValidation = DataName (test(crossval),:);
	Partition data into sets for training and validation. Partition the data into a training partition and a held-out partition for validation and testing. Specify the holdout percentage to be 30%.

c. Extract the text data	**Descriptions**

textDataTrain = dataTrain.TextVariableName_;
textDataValidation = dataValidation.
TextVariableName _;
YTrain = dataTrain. TextVariableName;
YValidation = dataValidation. TextVariableName;
	Extract the text data and labels from the partitioned tables. Here TextVariableName is column name for the group variable (for example age group) and TextVariableName is the column name for the text variable (in this example the name of cancer types)

d. Create a cloud for the text	**Descriptions**
figure wordcloud(textDataTrain);	To check that you have imported the data correctly, visualize the training text data using a word cloud.
e. Preprocess Text Data	

documentsTrain = preprocessText
(textDataTrain);
documentsValidation = preprocessText
(textDataValidation);
	Preprocess the training data and the validation data using the preprocessText function.

f. Convert document to sequences	**Descriptions**

enc = wordEncoding(documentsTrain);
sequenceLength = 10;
XTrain = doc2sequence(enc,
documentsTrain,'Length',sequenceLength);
XTrain(1:5)
XValidation = doc2sequence(enc,
documentsValidation,'Length',sequenceLength);
	Encoding to convert the documents into sequences of numeric indices.

g. Create and Train LSTM	**Descriptions**

inputSize = 1;
embeddingDimension = 30;
numHiddenUnits = 200;
numWords = enc.NumWords;
numClasses = numel(categories(YTrain));
layers = [...
sequenceInputLayer(inputSize)
wordEmbeddingLayer(embeddingDimension,
numWords)
lstmLayer(numHiddenUnits,'OutputMode','last')
fullyConnectedLayer(numClasses)
softmaxLayer
classificationLayer]
	Initialize the embedding weights

h. Specify training options	**Descriptions**
options = trainingOptions('adam', ... 'MiniBatchSize',30, ...	MiniBatchSize: Classifies data using mini batches of size

'GradientThreshold',8, ... 'Shuffle','every-epoch', ... 'ValidationData',{XValidation,YValidation}, ... 'Plots','training-progress', ... 'Verbose',false);	'GradientThreshold: Clip gradient values for the threshold
i. Train the LSTM network	**Descriptions**
net = trainNetwork(XTrain,YTrain,layers, options);	Train the LSTM network using the trainNetwork function.
j. Predict using new data	**Descriptions**
reportsNew = [... "The text definition here."];	Classify the event type of three new reports. Create a string array containing the new reports.
k. Preprocess Convert and Classify	**Descriptions**

documentsNew = preprocessText(reportsNew);
XNew = doc2sequence(enc, documentsNew,'Length',sequenceLength);
labelsNew = classify(net,XNew)	

All descriptions are based on MATLAB 2020b and related examples from the MATLAB.	

Table 2.
MATLAB codes and specification of cods for Example 2.*

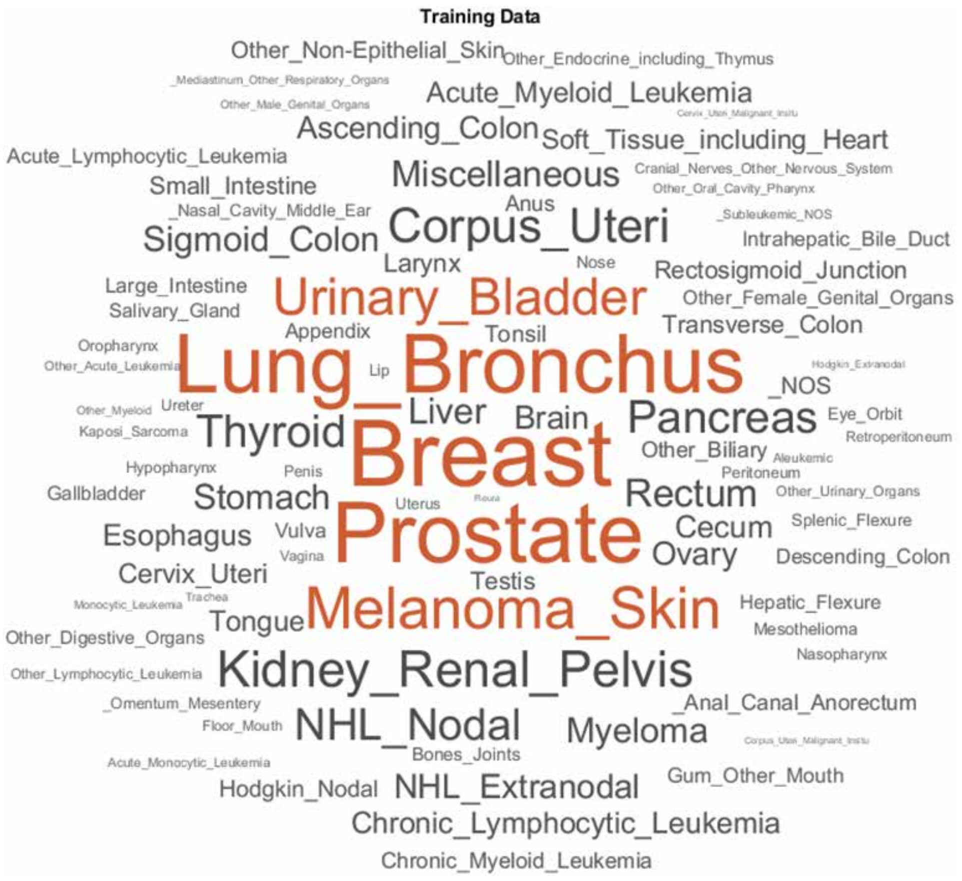


Figure 12.
Visualizing the training data text file for SEER-2017 cancer types by age groups using a word cloud of LSTM. The MATLAB codes to create this figure are given in ***Table 2****. The bigger the word, the more often diagnosed cancer type in 2017.*

age groups is illustrated in **Figure 12**. Here, in order to input the documents into an LSTM network, the "*wordEncoding(documentsTrain)*" is used. This code converts the name of diseases into sequences of numeric indices. The disease names (all types of text structure) in LSTM with MATLAB are performed in three consecutive steps: 1) tokenize the text 2) convert the text to lowercase and, 3) erase the punctuation. The function stops predicting when the network predicts the end-of-text character or when the generated text is 500 characters long.

5. Conclusions and future work

LSTM is a very powerful ANN architecture for disease subtypes, time series analyses, for the text generation, handwriting recognition, music generation, language translation, image captioning process. The LSTM approach is effective to make predictions as equal attention is provided for all input sequences by the information flows through the cell state. Because of the mechanism adopted, the small change in the input sequence does not harm the prediction accuracy done by LSTM. Future work on LSTM has several directions. Most LSTM architectures are designed to handle data evenly distributed between elapsed times (days, months, years, etc.) for the consecutive elements of a sequence. More studies are needed to improve the predictive ability of LSTM for nonconstant consecutive observations elapsed times. Moreover, further studies are needed for possible overfitting problems for training with smaller data sets. Rather than using early stopping to avoid the overfitting, Bayesian regularized approach would be more effective to ensure that the neural network halts training at the point where further training would result in overfitting. As the Bayesian regularized approach uses a different loss function with different hyperparameters, this approach demands costly computation resources.

Acknowledgements

The author would like to sincerely thank to Rosey Zackula for reading and revising the manuscript carefully.

Author details

Hayrettin Okut
The University of Kansas School of Medicine, Wichita, USA

*Address all correspondence to: hokut@kumc.edu

References

[1] Okut, H., Wu, X-L., Rosa, JM. G., Bauck, S., Woodward, B., Schnabel, D. R., Taylor, F. J. and Gainola, D. Predicting expected progeny difference for marbling score in Angus cattle using artificial neural networks and Bayesian regression models. Genetics Selection Evolution 2013, 45:34 doi:10.1186/ 1297-9686-45-34.

[2] Okut H.,. Bayesian Regularized Neural Networks for Small n Big p Data, Artificial Neural Networks - Models and Applications, Joao Luis G. Rosa, IntechOpen, 2016. DOI: 10.5772/63256.

[3] Hochreiterand, S. and Schmidhuber, J., Long Short-Term Memory. Neural Computation. Volume 9 | Issue 8, 1997

[4] Schmidhuber, J. Deep Learning in Neural Networks: An Overview". Neural Networks.61: 85 17, 2015. arXiv: 1404.7828.

[5] Miotto, R., et al., "Deep patient: An unsupervised representation to predict the future of patients from the electronic health records," Sci. Rep., vol.6, no. 1, pp. 26094–26094, 2016.

[6] Choi, E., et al., "Doctor AI: Predicting clinical events via recurrent neural networks," in Proc. 1st Mach. Learn. Healthcare Conf., 2016, pp. 301–318.t

[7] Razavian, N., J. Marcus, and D. Sontag, "Multi-task prediction of disease onsets from longitudinal lab tests," in Proc. 1st Mach. Learn. Healthcare Conf., 2016, pp. 73–100.

[8] Yang Chao-Tung, Yuan-An, C.., Wei Chan, Y., Chia-Lin L., Yu-Tse T., Wei-Cheng C. and· Po-Yu, L. Liu (2020). Influenza-like illness prediction using a long short-term memory deep learning model with multiple open data sources. The Journal of Supercomputing (2020) 76:9303–9329 https://doi.org/10.1007/ s11227-020-03182-5.

[9] S. Purushotham et al., "Benchmark of deep learning models on large healthcare mimic datasets," 2017.online available: h ttps://arxiv.org/abs/ 1710.08531

[10] Kim et al.,J. Y., "High risk prediction from electronic medical records via deep attention networks," Nov. 30, 2017. [Online]. Available: https: //arxiv.org/abs/1712.00010

[11] Ma, F., et al., "Dipole: Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks," in Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Halifax, Canada, 2017, pp. 1903–1911.

[12] Nguyen, P., Tran, T. and Venkatesh, S. "Resset: A recurrent model for sequence of sets with applications to electronic medical records," in Proc. Int. Joint Conf. Neural Netw., Brazil, 2018, pp. 1–9.

[13] Maxwell, A., et al., "Deep learning architectures for multi-label classification of intelligent health risk prediction," BMC Bioinf., vol. 18, no. Suppl 14, pp. 523–523, 2017.

[14] Tingyan Wang, Yuanxin Tian , and Robin G. Qiu. Long Short-Term Memory Recurrent Neural Networks for Multiple Diseases Risk Prediction by Leveraging Longitudinal Medical Records. EEE Journal Of Biomedical And Health Informatics, Vol. 24, No. 8, August 2020 DO:1 0.1109/ JBHI.2019.2962366.

[15] Baytas, I., Xiao, C., Zhang, X., Wang, F., Jain, K. A. and Zhou, Jiayu. Patient Subtyping via Time-Aware LSTM Networks. In Proceedings of KDD Halifax, NS, Canada, 2017..DOI: 10.1145/3097983.3097997.

[16] Okut, H., Gianola, D., Rosa, J. G., Weigel, K. Prediction of body mass

index in mice using dense molecular markers and a regularized neural network. Genetics Research (Cambridge). 2011. 93:189–201.

[17] Lipton, C. Z., Berkowitz, J. and Elkan, C. A Critical Review of Recurrent Neural Networks for Sequence Learning. arXiv:1506.00019v4.

[18] Colah, C. Understating LSTM Network. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[19] Ali. M. A., Zhuang, H., Ibrahim, A., Rehman, O., Huang, M and Wu, A. A Machine Learning Approach for the Classification of Kidney Cancer Subtypes Using miRNA. Genome Data. Appl. Sci. 2018, 8, 2422; doi:10.3390/app8122422.

[20] https://www.geeksforgeeks.org/lstm-derivation-of-back-propagation-through-time/?ref=lbp. 2020.

[21] Gers, F. A., Schmidhuber, J. and Cummins, F. Learning to forget: Continual prediction with LSTM. In Proc. ICANN'99, Int. Conf. on Artificial Neural Networks, Vol. 2, pp. 850–855, 2000. Edinburgh, Scotland. IEE, London. Extended version submitted to Neural Computation.

[22] Kyunghyun, C., van Merrienboer, Gulcehre, Caglar, F., Dzmitry, B., Fethi B.,Holger, H. and Yoshua, B. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, 2014.arXiv: 1406.1078.

[23] Krause, B., Murray, I. and Renals S. Multiplicative LSTM for sequence modelling., 2017. arXiv:1609.07959v3

[24] Wu, Y., Schuster,M., Chen, Z., Le V. Q., Norouzi, M., Macherey, W., Krikun, M, Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Taku, K., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M. and Dean, J. Google's Neural Machine Translation System: Bridging tshe Gap between Human and Machine. Translation.2017, arXiv:1609.08144v2

Section 2

Applications

Chapter 5

Modeling the Behavior of Amphiphilic Aqueous Solutions

Gonzalo Astray Dopazo, Cecilia Martínez-Castillo, Manuel Alonso-Ferrer and Juan Carlos Mejuto

Abstract

Two types of predictive models based on artificial neural networks (ANN) and quadratic regression model developed in our laboratory will be summarized in this book chapter. Both models were developed to predict the density, speed of sound, kinematic viscosity and surface tension of amphiphilic aqueous solutions. These models were developed taking into account the concentration, the number of carbons and the molecular weight values. The experimental data were compiled from literature and included different surfactants: i) hexyl, ii) octyl, iii) decyl, iv) tetradecyl and v) octadecyl trimethyl ammonium bromide. Neural models present better adjustment values, with R^2 values above 0.902 and AAPD values under 2.93% (for all data), than the quadratic regression models. Finally, it is concluded that the quadratic regression and the neural models can be powerful prediction tools for the physical properties of surfactants aqueous solutions.

Keywords: amphiphiles, surfactants, physical properties, modeling, artificial neural network

1. Introduction

Amphiphilic compounds have a well-defined structure; two parts clearly differentiated that will determine the behavior in aqueous systems [1] and is the key factor to their relationship with the internal and the external interfaces in aqueous systems [1]. One part of the amphiphilic compound is hydrophilic and the other part is hydrophobic [1, 2] and both are linked by a covalent bond [2].

In aqueous systems, the most important application of surfactants (in volume and economic impact terms), generally a long-chain hydrocarbon group is used as the hydrophobic group (although i) fluorinated, ii) oxygenated hydrocarbon or iii) siloxane chains can also be used) and an ionic or highly polar group as a head or hydrophilic group [3]. The different types of amphiphilic molecules can be differentiated according to the bonds between their two parts, hydrophilic and hydrophobic [2]. For example i) a hydrophilic head can be covalently bound to hydrophobic alkyl chain, whether single, double, or triple, also, ii) an amphiphilic bolaform is formed by two hydrophilic heads covalently linked with a hydrophobic alkyl chain and iii) a Gemini amphiphile is two surfactants covalently linked by their charged heads [2]. These compounds can be also classified based on the chemical nature of their hydrophilic group with subgroups according to the tail, so that, four basic categories can be defined: i) anionic, ii) cationic, iii) nonionic and iv) amphoteric (and zwitterionic) [3].

The property of amphiphiles to self-assemble in aqueous solution to design well-defined structures makes them become interesting molecules that can be applied in different fields [2] such as:

i. Pharmaceutical to overcome: i) the important manufacturing costs, ii) the poor pharmacokinetic characteristics and iii) the low bacteriological efficiency of the natural cationic antimicrobial peptides (AMPs), using novel and diverse cationic amphiphiles that can mimic the AMPs amphiphilic topology [4], or even as anti-cancer drug delivery vehicles using block copolymer micelles (poly(ethylene oxide) and poly(L-amino acid)) [5],

ii. in the cleaning sector, where they were used to clean oily deposits from solid surfaces using mixed solutions of fatty acid sulfonated methyl esters and using as cosurfactant dodecyldimethylamine oxide [6]. Yavrukova et al. [6] study the cleaning process of porcelain and stainless steel and concluded that the SME mixtures can be a hopeful system for formulations in household detergency,

iii. in Chemistry, where this kind of molecules are studied as a developer of supramolecular nanotubes architectures [7],

iv. in Medical Science to accelerate wound healing using antioxidant shape amphiphiles [8], or

v. in Food Chemistry using amphiphiles to modulate organoleptic properties in foods post harvested technology or for potential food applications [9, 10], among others.

As previously said, these kinds of molecules can form different types of aggregates. These structures are formed when a certain concentration, called critical micelle concentration (cmc), is reached. This parameter can be defined as the specific concentration for a particular surfactant at which determinate solution properties change strongly [3]. According to Myers [3], different authors showed that the aggregated structure type depends on what is known as critical packing parameter. This parameter ($CPP = v/a_o l_c$) establishes the relationship between the volume of the hydrophobic part of the molecule (v), the optimal area of the head group (a_o) and the critical length of the hydrophobic tail (l_c), and it controls the packing in aggregate structures [3]. The structures that can be formed are i) spherical micelles (when the value of CPP is less than 1/3), ii) cylindrical micelles ($1/3 < CPP < 1/2$), iii) bilayer vesicles ($1/2 < CPP < 1$), iv) lamellar phases ($CPP \approx 1$) and, finally, v) inverted micelles ($CPP > 1$) [11]. Some of these structures are shown in **Figure 1**.

According to Gómez-Diaz et al. [1, 12], different physical properties have been used to characterize the aggregation processes by means of measured different experimental values. These authors have been demonstrate that density and kinematic viscosity do not alter when the micellization point is reached so that they are not utilized to determine knowledge about the behavior of the colloidal aggregate [1, 12]. On the other hand, the variation of the rest of the measured properties, speed of sound and surface tension, can be used to determine the cmc value. The property variation can give rise to the existence of two trend lines which intersection can be used to determine the cmc [1, 12]. As claimed by Gómez-Díaz et al. [1, 12], the cmc value, using the surface tension and the speed of sound was similar. Nevertheless, the cmc value using the surface tension was, for the hexyl, octyl and

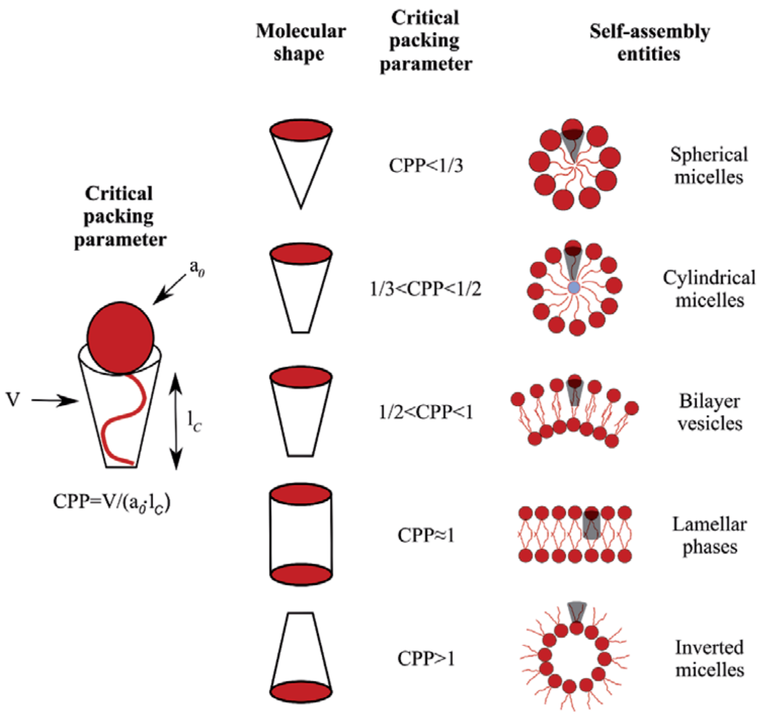


Figure 1.
Critical packing parameter (where: v is the lipophilic chain volume; a_0 is the hydrophilic core cross-section area and l_c is the lipophilic chain length), molecular shape and self-assembly entity formed by each different amphiphile. Redraw from the original figure of Wang et al. [11].

decyl trimethyl ammonium bromide, a bit lower than when the speed of sound is used [1, 12] (which can be attributed to the effect of small impurities amounts upon the surface tension value) [1].

The study of solutions behavior to know its properties, and to be able to calculate the cmc, required a lot of work, time-consuming and material cost. Due to these facts, modeling the physical properties of these solutions could help to reduce material and time costs. Thus, the study of methodologies such as artificial neural networks (ANN) and response surface (RS) are interesting and due to this in our research group, a study about this possibility were carried out by Astray & Mejuto [13].

On the one hand, and regarding response surface methodology, it was firstly described by Box and Wilson in 1951 [13, 14]. The RSM is used as a tool for optimization tasks by relating the variables of the process and its response [15, 16]. The experimental data could be fit to a polynomial equation which must describe the data behavior to achieve statistical previsions [17], therefore, this methodology is based on the development of empirical mathematical models to describe the system under study [18]. These models can be used when the response, or responses, are influenced by different variables [17]. An RSM model can work with a reduced amount of experimental trials and can be used to develop, improve and optimize different process [19]. The RSM can use a set of mathematical and statistical tools to fit the experimental data to an Equation [17], usually, linear or square polynomial functions [17, 18]. Different experimental designs could be used which randomizes the experimental error and equals the experimental points distribution, for de independent variables, in the range investigated [20]. RSM models can be applied in different areas such as:

i. Chemical Engineering to extract alumina from coal fly ash optimizing different variables involved in the process ($K_2S_2O_7/Al_2O_3$ molar ratio, calcining temperature and calcining time) [21],

ii. in Environmental Science to study the biodegradation of the strobilurin fungicide Pyraclostrobin using bacteria from orange cultivation plots to develop a bioremediation method [22],

iii. in Biomedical applications to extract anthocyanins from blueberry optimizing the ultrasonic time, ultrasonic temperature, freezing time and liquid–solid ratio [23] or in

iv. in Biotechnology to optimize the culture media and reduce the production cost of urease bacteria to achieve an eco-friendly process controlling different parameters (yeast extract, whey and heating temperature) [24], inter alia.

On the other hand, artificial neural networks are computational modeling tool that consists of a set of simple processing elements (neurons), massively interconnected capable to process data [20]. This kind of models can try to simulate the path in which the human brain process the information, that it is, ANN are inspired in the biological system [25]. ANN is made up of different neurons layers: an input layer to receive the information, one or more intermediate (or hidden) layers where the information is processed, and an output layer, with one or more neurons, where the predicted value is generated (**Figure 2**). Each neural network is characterized by a specific topology or architecture. To facilitate your identification each neural model implemented can be named such as i-h-o, using the number of neurons presented in the input (i), hidden (h) and output (o) layer [13].

These models present different advantages such as: are non-linearity systems that allow better data fit, are non-sensitivity to noise (uncertain data and measurement errors), present high parallelism (fast processing and failure-tolerance), among others [26]. According to Baş and Boyaci [20], ANNs represent non-linearities better than RS, although ANNs cannot produce a similar model equation to RS models. This kind of approach can be used in a multitude of fields such as:

i. Engineering to diagnose and classify of bearing faults [27] or to model hot deformation in titanium alloys [28],

ii. in Food Technology to determine the botanical origin of honey using different parameters (ashes content, electrical conductivity, among others) [29] or food authenticity [30] (carried out in our laboratory),

iii. in Renewable Energy to predict three components of solar irradiation in Odeillo (France) [31], or

iv. or in diverse fields (also carried out in our laboratory) such as Palynology [32] or Hydrology [33], among others.

This book chapter summary the quadratic regression and neural models developed in our research group [13] to predict, for amphiphilic aqueous solutions, the i) density (ρ in $g{\cdot}cm^{-3}$), ii) speed of sound (u in $m{\cdot}s^{-1}$), iii) kinematic viscosity (ν in $mm^2{\cdot}s^{-1}$), and iv) surface tension (σ in $mN{\cdot}m^{-1}$) taking into account i) concentration (C), ii) carbons number (n °C) and iii) molecular weight (M_w).

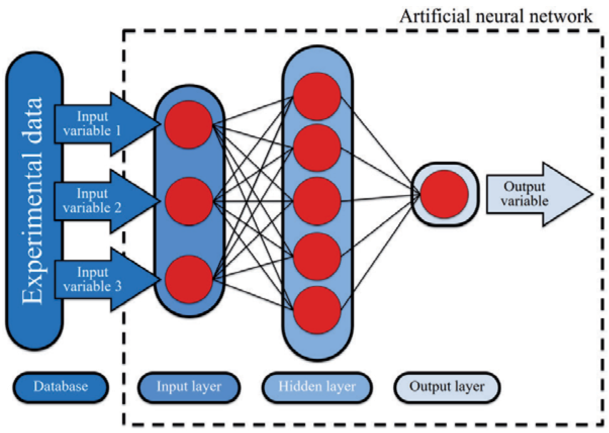


Figure 2.
Representation of a typical neural network with 3 neurons in the input layer, 5 neurons in one hidden layer and one neuron in the output layer (topology 3–5-1). Redraw from the original figure of Astray & Mejuto [13].

2. Material and methods

2.1 Artificial neural networks as an approximation approach

Artificial Intelligence models based on artificial neural networks have been widely used in the area of chemistry to model and predict processes related to physical properties. This type of model has shown great reliability to model and predict density, dynamic viscosity, and surface tension, among others.

A good example of the use of artificial neural networks to determine properties of interest in micellar systems is the research carried out by Katritzky et al. [34] who developed a model to predict the critical micellar concentration of non-ionic surfactants based on different parameters related to its molecular structure. According to the authors, the models developed could be used for prediction or analysis of new non-ionic surfactants similar to those used in this research. On the other hand, Fatemi et al. [35] developed a model based on artificial neural networks to predict the critical micellar concentration of different anionic and cationic compounds. The selected input variables included the Balaban index, the heat of formation, among others. The results obtained were compared with the predictions of a multiple linear regression model and it was shown that the neural network is superior to multiple linear regression model to predict the log CMC of anionic and cationic surfactants. Along the same line, Kardanpour et al. [36] reported a wavelet neural network (WNN) to predict the critical micellar concentration of Gemini surfactants. The developed model used twelve different descriptors from the molecular structure. According to the authors, the results reveal the ability of the model to determine CMC and demonstrate, in comparison with MLR models, that the models based on neural networks are superior to the MLR approach (due to the ability of the WNN model to work with nonlinearities between the input variables and the CMC).

The researchs listed above demonstrate the ability of artificial neural networks to predict the critical micellar concentration of different surfactants. But to predict the value of this CMC, it is necessary to carry out different experimental studies to determine any particular property that allows determining the CMC value as a function of some abrupt variation of that property. Two of these properties are surface tension and speed of sound whose experimental work requires a great deal of work time and expense in labour and reagents. The different experiments carried out for each variable would determine the CMC as a function of the intersection of the two trend lines (as mentioned above [1, 12]). Due to these facts, an ANN approach could be very useful to lower costs and be able to make approximations easier, so designing models that are capable to predict this variable depending on the different mixtures could be a very recommendable tool. The claim that artificial neural networks are useful tools because they can minimize the time of experimental treatment and operating costs can be contrasted in different studies reported in the bibliography. An example of this, is the study carried out by Belhaj et al. [37] in which they use artificial neural networks to predict absorption values for alkyl ether carboxylate (AEC) and alkyl polyglucoside (APG). Thus, this book chapter summary the research carried out in our research group to predict density, speed of sound, kinematic viscosity and surface tension of amphiphilic aqueous solutions [13].

In addition to our work, surface tension modeling was also carried out by different authors. Khazaei et al. [38] developed an ANN to predict the surface tension of multicomponent mixtures at different temperatures was employed. The input variables were: reduced temperature, critical pressure and volume, and an acentric factor of the mixture. The obtained average absolute relative deviations were low and the ANN model, compared with well-known models (Brok-Bird equation, Flory theory and group contribution theory) has proved a high prediction capacity. The authors concluded that ANN can be helpful for engineering calculations and they emphasized that the ANN model can be a robust approach to predict complex input–output systems. Other interesting research was carried out by Gharagheizi et al. [39] developed neural models to determine the surface tension of pure compounds at different temperatures and atmospheric pressure. The authors investigated compounds belonging to 78 different chemical families and the results were satisfactory (according to different statistical parameters) with an absolute average deviation of 1.7% and a squared correlation coefficient of 0.997. On the other hand, Bakeri et al. [40] used 20 hydrocarbons mixtures to determine the surface tension. The model developed by the authors showed the best accuracy when they are compared with other four well-known classical models. On the other hand, density and kinematic viscosity, in this case, for different systems of biofuels and their blends with diesel fuel, can be predicted using ANNs [41]. In this case, two artificial neural networks were developed to predict kinematic density and viscosity. The models developed used 6 input variables, temperature, volume fractions, among others. The results reported by the authors indicate that the models obtained good correlations. Density and speed of sound of binary ionic liquid and ketone mixtures can also be predicted by ANNs [42]. In this case, the artificial neural network models used as input variables, the temperature and the mole fraction, among others, to determine these two variables. The models developed presented an overall average percentage error lower than 2.5%, so the authors concluded that this model was applicable for the prediction of these variables in binary ionic liquid and ketone mixtures.

Nevertheless, the use of artificial neural networks is not only limited to the prediction of the previous properties, ANNs can also be used to tensammetric analysis of different nonionic surfactants (Brij 30, 35, 56 and 96) [43]. Authors concluded that ANNs can be a possible candidate to determine nonionic surfactants. Another interesting study is the one developed by Jha et al. [44] that developed a feedforward artificial neural network with three layers to predict the diffusion coefficient of a micellar system with sodium dodecyl sulfate (SDS). The model uses the temperature and NaCl and SDS concentrations as input variables. The ANN is capable to model the experimental behavior (correlation coefficient upper than 0.99) and it is concluded that the model is usable to calculate this property. ANN models can also be used to investigate the different factors that affect particle size in a Nanoemulsion System (Virgin Coconut Oil) that contain copper peptide [45]. The model used, to predict the particle size, four input variables composed of the amount of virgin coconut oil, Tween 80:Pluronic F68, xanthan gum and water. The ANN demonstrated its ability to model the particle size according to the four input variables and showed good determination coefficients upper than 0.97. Finally, another interesting research is that carried out by Rocabruno-Valdés et al. [46] in which the authors develop artificial neural models to predict different properties (dynamic viscosity, density and cetane number of biodiesel) using as input variables the temperature, the number of carbon and hydrogen atoms and methyl esters composition. The correlation coefficients obtained were upper than 0.91. According to the authors, the ANN models provide an adequate prediction and can be interesting for their inclusion in simulators.

2.2 Database

To carry out this work, the experimental data obtained by Gomez et al. [1, 12] were used. The used surfactants were: hexyl trimethyl ammonium bromide (HTABr), octyl trimethyl ammonium bromide (OTABr) and decyl trimethyl ammonium bromide (DTABr) from [1] and tetradecyl trimethyl ammonium bromide (TDTABr) and octadecyl trimethyl ammonium bromide (ODTABr) from [12]. All these reagents were supplied by Fluka with a purity ≥98%. [1, 12]. The authors prepared the aqueous solutions by mass using an analytical balance Kern 770 (precision 10^{-4} g) [1, 12].

The output variables for each aqueous solution were determined (at 298 K) with different instruments: i-ii) the density and speed of sound using an Anton Paar DSA 5000 vibrating-tube densimeter and sound analyzer, iii) the kinematic viscosity by means the transit time for liquid meniscus through a capillary viscosimeter (supplied by Schott) and iv) the surface tension using a tensiometer Krüss K-11 using the Wilhelmy plate method [1, 12].

2.3 Modeling procedure for predictive models

The surface model, which is used to evaluate the influence of each input variable on the physical properties (density, speed of sound, kinematic viscosity and surface tension) used the combination of input variables linearly, quadratically and cross-correlated [13]. That it is, experimental data can be approximate using a generalized second-order polynomial model (Eq. (1) [47]). In this sense, the model was used to correlated each dependent variable (y_{pred}) using the input variables (x), the regression coefficients (b) and the random error value (ε).

$$y_{pred} = b_0 + \sum_{j=1}^{k} b_j x_j + \sum_{j=1}^{k}\sum_{i=1}^{k} b_{ji} x_j x_i + \sum_{j=1}^{k} b_{jj} x_j^2 + \varepsilon \qquad (1)$$

The response surface methodology was created to carry out the experiments with previous analysis of the relationship between the variables (generally standardized), with a homogeneous distribution of the experiments [13]. Nevertheless, in this case, the experimental data used are not homogeneously distributed and the data have not been standardized [13].

The other predictive model used is based on artificial neural networks. The ANN require to split the data into at least two different groups -training (T) and validation (V)-, which has been carried out by the authors [13] randomly. The set of training data was used to train the ANN model, while the validation data set is used to check the good training of the model [13]. An important aspect of this methodology is that it based on the trial-error procedure [13] to find the optimal combination of parameters for prediction. Once the database is presented to the input layer, the training can start, the data are propagated to the first intermediate layer and the information is treated by the propagation function (Eq. (2)) to obtain a single value (S_i), being: x_f the input data (in the input neuron *f*), w_{fi} the weight (importance among the neurons *f* and *i*) and b_i the bias value associated with the intermediate neuron *i*. [13]. The single response is processed by the activation function (Eq. (3)) being: S_{res} the output value of the neuron [13]. This process is repeated throughout all the neurons in the intermediate and output layer where the predicted value is generated.

$$S_i = \sum_{f=1}^{N} w_{fi} x_f + b_i \tag{2}$$

$$S_{res} = \frac{1}{1+e^{-S_i}} \tag{3}$$

2.4 ANN's parameters

The authors [13] used a total of 80 cases to develop different prediction models (RS and ANN). In this case, the database was divided into two groups. A first group, with 75% of the cases (60), to train the model and a second group, with the remaining 25%, to validate the model (20) [13].

The learning rate and momentum values were set at 0.7 and 0.8, respectively. The models were developed at different training cycles in order to locate the point from which could be overtrained.

2.5 Adjustments parameters

The results were analyzed by the authors [13] using different statistics to determine the adjustment power, such as the coefficient of determination (R^2), the root mean square error (RMSE) (Eq. (4)) or the average absolute percentage deviation (AAPD) (Eq. (5)) for the training and the validation phases. Individual percentage deviations (IPD) is also used.

$$RMSE = \sqrt{\frac{\sum_{j=1}^{n}\left(y_{pred} - y_{experimental}\right)^2}{n}} \tag{4}$$

$$AAPD = \frac{\sum_{j=1}^{n}\left(\left|\frac{y_{pred} - y_{experimental}}{y_{experimental}}\right|\right) \cdot 100}{n} \tag{5}$$

2.6 Computer equipment and software

The input variables, necessary to determine the desired variables, were obtained from the Sigma Aldrich and Chemdraw Professional 15 trial (PerkinElmer) [13]. Microsoft Excel Professional Plus 2013 (Microsoft) was used for RS modeling, and the software EasyNN plus v14.0d (Neural Planner Software Ltd.) was used to ANN modeling [13]. A computer server with an Intel® processor Core™ i7 processor with 16 GB of RAM was used to develop the models [13].

The figures of this book chapter were made with Inkscape 0.92 and Microsoft PowerPoint Professional Plus 2016 (Microsoft).

3. Results and discussion

The adjustments for the models developed [13] are shown in **Table 1**. It can be seen heterogeneous results for the surface and neural models.

Response surface models present good determination coefficients in the training phase, varying between the value obtained for the density model (0.994) and the value obtained for the kinematic viscosity model (0.906). These good values contrast with the value obtained for the surface tension model which reports a low determination coefficient value (0.505).

For the first three models (density, speed of sound and kinematic viscosity) the values of determination coefficient obtained in the validation phase are similar (with a minimal descent to the obtained R^2 values in the training phase) varying between the value obtained for the density model (0.985) and the R^2 value obtained for the kinematic viscosity model (0.885). The response surface model, with the worst-performing behavior for the training phase, the model developed to predict surface tension, showed, for the validation phase, a determination coefficient of 0.503, similar to that obtained in the training phase (0.505).

Regarding the root mean square error values obtained by the response surface models developed by Astray & Mejuto [13], it can be seen that the density model present an RMSE value around 0.001 $g{\cdot}cm^{-3}$, in both phases, the speed of sound models around 7.1837 $m{\cdot}s^{-1}$ and 6.3226 $m{\cdot}s^{-1}$ in training and validation phase, respectively. The model developed to predict kinematic viscosity presents an RMSE value around 0.1003 $mm^2{\cdot}s^{-1}$ for the training phase and 0.0569 $mm^2{\cdot}s^{-1}$ for validation phase, and the worst model developed, the surface tension model, 8.3304 $mN{\cdot}m^{-1}$ and 8.1307 $mN{\cdot}m^{-1}$, for training and validation phase, respectively. The size of these errors can best be understood if they are given in terms of average absolute percentage deviation. The AAPD values reported for each phase are very similar. In this case, the errors obtained for each model (for all data) were: 0.08%, 0.31%,

	Training phase		Validation phase	
Model	**R^2**	**RMSE**	**R^2**	**RMSE**
RS_ρ	0.994	0.0012	0.985	0.0011
RS_u	0.976	7.1837	0.972	6.3226
RS_ν	0.906	0.1003	0.885	0.0569
RS_σ	0.505	8.3304	0.503	8.1307
ANN_ρ	0.999	0.0004	0.999	0.0003
ANN_u	0.998	1.9393	0.998	1.7093
ANN_ν	0.999	0.0108	0.994	0.0104
ANN_σ	0.449	9.6859	0.457	9.6827
ANN'_σ	0.985	1.4956	0.940	2.8593

Table 1.
Adjustments for training and validation phase for the RS and ANN models selected by the authors [13]. Determination coefficient (R^2) and root mean square error (RMSE) for the models developed by surface (RS) and neural models (ANN). The subscript ρ corresponds to the variable density, u to the speed of sound, ν to the kinematic viscosity and σ is the surface tension. Table adapted from data reported by Astray & Mejuto [13].

5.18% and 14.73%, for density, speed of sound, kinematic viscosity and surface tension model, respectively. It can be seen how the AAPD value for the density and speed of sound prediction models are very low, the error of the kinematic viscosity model presents an error of 5.18% that can be considered feasible. In these cases, the error that is not acceptable is the one reported by the surface tension model (14.73%) since it is clearly much higher than the rest, and above the 10% which is considered, in our laboratory, as an acceptable error.

With all this, it can be said that the models designed to determine the density, the speed of sound and the kinematic viscosity are useful models for the prediction of these properties. The model to predict the surface tension should not be used due to its high APPD.

The adjustments for the ANN models developed [13] can be shown in **Table 1**. ANN models were developed based on the trial-error method to obtain the best models for each predict output variable (more than 400 neural networks were developed) [13]. All models developed by the authors [13] presented a different topology: i) 3–7-1 for the density model, ii) 3–5-1 for the speed of sound model, iii) 3–6-1 to the kinematic viscosity model and iv) 3–1-1 for the surface tension model. Thus, each model presents, in the input layer, three variables: concentration, number of carbons and molecular weight and intermediate layer of each model varies from a single neuron, to predict the surface tension, to seven in the density model, in addition to that, each selected model has a different number of training cycles [13].

It can be observed (**Table 1**) that, in general, the ANN provided by the authors [13] adjust, properly, the desired variables, both in the training and in the validation phase. The model to predict the density value is the model with the best adjustments, in fact, and take into account the adjustments in terms of determination coefficient and root mean square error, this model presents values of 0.999 and 0.0004 $g{\cdot}cm^{-3}$, respectively, for the training phase and values of 0.999 and 0.0003 $g{\cdot}cm^{-3}$, respectively, for validation phase. Once again, as was RS models case, the model to predict the density is the model with the best adjustments, in fact, the AAPD values reported for both phases were 0.02%.

The behavior of the rest of the models follows the pattern of the RS models, that is, the models to predict the speed of sound and the kinematic viscosity are, in this order, the second and the third-best model [13].

The model destined to predict the speed of sound presents adjustments, in terms of coefficient of determination, very close to the model destined to predict density (0.998 in both phases), presenting relatively low RMSE values (1.9393 $m{\cdot}s^{-1}$ and 1.7093 $m{\cdot}s^{-1}$).

The kinematic viscosity model has slightly lower adjustment than the previous two models. In this sense, and always in terms of the determination coefficient, the value for the training phase remains similar to the two previous models, however, for the validation phase, this value falls slightly to 0.994. Even so, the model seems to be predicting the kinematic viscosity values correctly, especially if it be taking into account the low RMSE values (0.0108 $mm^2{\cdot}s^{-1}$ and 0.0104 $mm^2{\cdot}s^{-1}$, for training and validation, respectively) [13].

Finally, the worst model developed using artificial neural networks is the model designed to determine surface tension [13]. It can be seen in **Table 1** show the values obtained fall significantly, in fact, the determination coefficient value falls to 0.449 and 0.457 for the training and validation phase, respectively. It seems clear that this low value of determination coefficient indicates the impossibility of the model to make correct predictions. This fact is demonstrated with the high RMSE values obtained for the training and validation phase (9.6859 $mN{\cdot}m^{-1}$ and 9.6827 $mN{\cdot}m^{-1}$, respectively).

As stated above, the size of the errors made by the different ANN models can best be understood in terms of AAPD. In this case, the errors obtained (for all data) by density, speed of sound, kinematic viscosity and surface tension model were: 0.02%, 0.10%, 0.62% and 18.13%, respectively.

In the same way that occurs with the surface models, the ANN surface tension model should not be used to predict surface tension (APPD above 10%). The other three models can be used for prediction.

3.1 Comparison of response surface and neural models

Once the models have been analyzed separately, it is necessary to make a comparison between them.

As previously stated, the models to predict density are the best models according to the adjustments. This means that this model is useful to predict physical properties of surfactants aqueous solutions (at least with the surfactants studied).

On the one hand, although in general, the AAPD in the RS_ρ model is around 0.08%, according to the authors [13], some cases present a bigger IPD value (0.25–0.49%). Even so, these values are very low. Despite the good performance of this RS model, the ANN model seems to work a little better, improving each adjustment parameter (see **Table 1**). In fact, the AAPD values in the case of the ANN_ρ are below to the value obtained by the RS_ρ model. This improvement is observable in terms of RMSE being, for both phases together (with all the data), very significate (0.0012 $g\cdot cm^{-3}$ vs. 0.0004 $g\cdot cm^{-3}$) which represents an important improvement. For both models, it seems clear that the most important variable to determine the density is the concentration with an importance value around 59.00% for the ANN_ρ model and around 89.25% for ANN_ρ model [13].

The second-best models, based on their adjustments, are the models to predict the speed of sound. The RS_u and ANN_u model developed by Astray & Mejuto [13], present good results, in fact, the RS_u model presents, for all data, an R^2 value of 0.974 (with some cases presenting an IPD >1%), while for the ANN_u presents a better value of determination coefficient (0.998), representing a slight improvement of 2.46%. The same behavior occurs regarding the RMSE, where the ANN_u model improved this parameter by around 73.00%. The authors [13] reported that the ANN model has an AAPD value of around 0.10% and a highest IPD value around 0.45%. In both cases, very similar values are obtained for the training and validation phases. Concerning the importance of the variables, in the same way, that the models developed to predict the density, the most important variable to determine the speed of sound is the concentration with an importance value of 89.31% for the RS_u model and 63.30% for the ANN_u model [13].

The third-best model according to its results is the model to predict the kinematic viscosity. In this case, the behavior of the RS_ν model is slightly different from the one presented by the ANN_ν model. Thus, it is observed that the RS_ν model cannot predict with accuracy the kinematic viscosity and showed a slight dispersion of the data (predicted vs. experimental) that can be seen in the figure presented by Astray & Mejuto [13]. This dispersion is reflected in the adjustment parameters of the RS_ν model that presents, for all cases, a determination coefficient of 0.903 and an AAPD value of 5.18%). It is noteworthy that, according to Astray & Mejuto [13], there are more than 30 cases with an IPD value in the range of 5.34% to 26.51%. On the other hand, the model based on ANN, predicts with accuracy for the training and the validation phase, showed an R^2 upper than 0.993. According to the AAPD values provided, for all data, the AAPD value obtained by the ANN_ν model (0.62%) compared to the AAPD value of the RS_ν model (5.18%) represents a decrease around 88.05% [13]. Regarding the input variables, in the same way, that the models developed to predict the density and the speed of sound, the most important variable to determine the kinematic viscosity is the concentration [13].

Finally, from all models developed, both surface tension models were the worst models according to their adjustments. These models are the models with the highest dispersion (RS_σ model can be seen in Figure 3.G [13]). The adjustment parameters for all data are low, in fact, the determination coefficient for the RS_σ model is around 0.503 (being even lower in the case of ANN_σ model –0.451-). Taking into account the AAPD values for all data, it can be seen how neither the RS_σ model (14.73%) and the ANN_σ model (18.13%) are capable to predict with accuracy the surface tension value. Regarding the input variables, once again, the most important variable to determine the surface tension is the concentration [13].

Due to these poor results, the authors [13] proposed an alternative ANN model (ANN'_σ) to improve the prediction of surface tension. In this new ANN model, the input variables were increased using the predictions of the ANN models of density, speed of sound and kinematic viscosity, that is, this new ANN model presents an input layer with six variables. The ANN'_σ model needs more cycles for its training (12800 cycles) compared to the three input variables ANN model (1200 cycles). It can be shown a notable improvement in the adjustments for the training and the validation phases. The determination coefficient increases to 0.985 for the training phase and 0.940 for the validation phase, while the RMSE values decrease from the 9.6851 $mN \cdot m^{-1}$ (for all case) to 1.9291 $mN \cdot m^{-1}$. In the same way, an improvement in the AAPD values is observed, which, in the new model, is below 3.86%. This new model takes the predicted speed of sound as the most important variable, unlike the previous ones, where the most important variable was focused on the concentration. The predicted speed of sound is followed by the predicted kinematic viscosity, the predicted density and the concentration (number of carbons and molecular weight present lower importance).

Given the results obtained by the surface models and the neural models [13], it can be concluded that the models developed to determine density, sound speed and kinematic viscosity are models suitable for their use in the laboratory due to the low APPD values that presented (between 0.02% and 5.18%, for all the data cases). Regarding the models for surface tension prediction, as previously mentioned, these cannot be used for laboratory use, because they present errors upper than 10%. The alternative ANN model developed by the authors [13], appears to offer acceptable results in terms of determination coefficient and AAPD value. This alternative model improves the original RS and ANN model.

All the models developed [13] can be improved in different ways. The response surface models could be improved by adapting the experimental cases to an experimental design before the experimental measurements, allowing on the one hand to save economic costs and time, and on the other, favoring the development of an RS model based on a precise experimental design. It would also be very convenient to develop a response surface model trying to find surfactants that allow the variables to vary constantly in a range. All these improvements could favor the improvement of the models destined to predict the density, the speed of the sound and the kinematic viscosity.

Likewise, and given the ANN model that uses six input variables [13], it would be interesting to develop an RS model that includes the predictions of density, speed of sound and kinematic viscosity as input variables of the model (although it would be necessary to see how to treat the different variation of the values within the range understudy).

Neural network models could be improved by including different input variables that are capable of better identification of the different surfactants. Another interesting approach could by the increase the database for their modeling.

4. Conclusion

The development of models based on response surfaces and neural networks to predict different physical properties of surfactants aqueous solutions (i) density, ii) speed of sound, iii) kinematic viscosity and iv) surface tension) can be a good alternative to save money and time in the laboratory.

In general terms, this kind of models can adjust, with accuracy, the density, the kinematic viscosity and the speed of sound with determination coefficient upper than 0.902 and lower APPD values than 5.20% (for all data). In contrast to these good adjustments, surface tension models do not work properly and presented (for all data) low determination coefficients (0.503 and 0.451 for RS and ANN model, respectively) and high APPD values (14.73% and 18.13% for RS and ANN model, respectively). It seems that this problem can be solved, in the case of models based on neural networks, with the inclusion of new variables from the predictions of the previous models. With this modification, the new neural model improves (for all data) each adjustment parameter (0.974 and 2.92% for determination coefficient and AAPD value, respectively).

In conclusion, RS and ANN models can be powerful prediction tools for the properties (density, speed of sound, kinematic viscosity or surface tension) of surfactants aqueous solutions. These models could therefore facilitate daily laboratory work, saving time and money. However, it would be interesting to improve the models using other development alternatives or, even, improve these model using different approaches such as support vector machines or random forests, among others.

Acknowledgements

Gonzalo Astray thanks to the University of Vigo for his contract supported by *"Programa de retención de talento investigador da Universidade de Vigo para o 2018"* budget application 0000 131H TAL 641. Cecilia Martínez-Castillo and Manuel Alonso-Ferrer thanks to the University of Vigo for her contract supported by FEADER 2018/002B project (Xunta de Galicia, Consellería de Medio Rural, Project "*Desarrollo de modelos de predicción de origen en vinos de denominaciones de origen gallegas*").

Author details

Gonzalo Astray Dopazo^{1,2*}, Cecilia Martínez-Castillo^{3}, Manuel Alonso-Ferrer^{3} and Juan Carlos Mejuto^{1}

1 Departamento de Química Física, Facultade de Ciencias, Universidade de Vigo, Ourense, España

2 CITACA, Universidade de Vigo Campus Auga, Ourense, España

3 Grupo de Nutrición y Bromatología, Departamento de Química Analítica y Alimentaria, Facultade de Ciencias, Universidade de Vigo, Ourense, España

*Address all correspondence to: gastray@uvigo.es

References

[1] Gómez-Díaz D, Navaza JM, Sanjurjo B. Density, kinematic viscosity, speed of sound, and surface tension of hexyl, octyl, and decyl trimethyl ammonium bromide aqueous solutions. J Chem Eng Data. 2007;52(3): 889-91.

[2] Zhang X, Wang C. Supramolecular amphiphiles. Chem Soc Rev. 2011;40(1):94-101.

[3] Myers D. Surfactant Science and Technology: Third Edition. Surfactant Science and Technology: Third Edition. 2006. 380 p.

[4] Findlay B, Zhanel GG, Schweizer F. Cationic amphiphiles, a new generation of antimicrobials inspired by the natural antimicrobial peptide scaffold. Antimicrob Agents Chemother. 2010;4049-58.

[5] Kwon GS, Kataoka K. Block copolymer micelles as long-circulating drug vehicles. Adv Drug Deliv Rev. 1995;16(2-3):295-309.

[6] Yavrukova VI, Shandurkov DN, Marinova KG, Kralchevsky PA, Ung YW, Petkov JT. Cleaning Ability of Mixed Solutions of Sulfonated Fatty Acid Methyl Esters. J Surfactants Deterg. 2020;23(3):617-27.

[7] Shimizu T, Masuda M, Minamikawa H. Supramolecular nanotube architectures based on amphiphilic molecules. Chem Rev. 2005;105(4):1401-43.

[8] Li Z, Zhang J, Fu Y, Yang L, Zhu F, Liu X, et al. Antioxidant shape amphiphiles for accelerated wound healing. J Mater Chem B. 2020 Aug;8(31):7018-23.

[9] Cid A, Morales J, Mejuto JC, Briz-Cid N, Rial-Otero R, Simal-Gándara J. Thermodynamics of sodium dodecyl sulphate-salicylic acid based micellar systems and their potential use in fruits postharvest. Food Chem. 2014 May;151:358-63.

[10] Cid A, Mejuto JC, Orellana PG, López-Fernández O, Rial-Otero R, Simal-Gandara J. Effects of ascorbic acid on the microstructure and properties of SDS micellar aggregates for potential food applications. Food Res Int [Internet]. 2013;50(1):143-8. Available from: http://www.sciencedirect.com/science/article/pii/S0963996912004206

[11] Wang N, Chen M, Wang T. Liposomes used as a vaccine adjuvant-delivery system: From basics to clinical immunization. J Control Release. 2019;303:130-50.

[12] Gómez-Díaz D, Navaza JM, Sanjurjo B. Density, kinematic viscosity, speed of sound, and surface tension of tetradecyl and octadecyl trimethyl ammonium bromide aqueous solutions. J Chem Eng Data. 2007;52(5):2091-3.

[13] Astray G, Mejuto JC. Approach of different properties of alkylammonium surfactants using artificial intelligence and response surface methodology. Tenside, Surfactants, Deterg. 2017;54(2):132-40.

[14] Box GEP, Wilson KB. On the Experimental Attainment of Optimum Conditions. Johnson NL, editor. J R Stat Soc Ser b. 1951;13(1):1-45.

[15] Yang QQ, Gan RY, Zhang D, Ge YY, Cheng LZ, Corke H. Optimization of kidney bean antioxidants using RSM & ANN and characterization of antioxidant profile by UPLC-QTOF-MS. LWT. 2019;114:108321.

[16] Natabirwa H, Nakimbugwe D, Lung'aho M, Muyonga JH. Optimization of Roba1 extrusion conditions and bean extrudate properties using response surface methodology and multi-response desirability function. LWT. 2018;96:411-8.

[17] Bezerra MA, Santelli RE, Oliveira EP, Villar LS, Escaleira LA. Response surface methodology (RSM) as a tool for optimization in analytical chemistry. Talanta. 2008;76(5):965-77.

[18] Teófilo RF, Ferreira MMC. Quimiometria II: planilhas eletrônicas para cálculos de planejamentos experimentais, um tutorial. Quim Nova. 2006;29(2):338-50.

[19] Sarkar M, Majumdar P. Application of response surface methodology for optimization of heavy metal biosorption using surfactant modified chitosan bead. Chem Eng J. 2011;175:376-87.

[20] Baş D, Boyaci IH. Modeling and optimization II: Comparison of estimation capabilities of response surface methodology with artificial neural networks in a biochemical reaction. J Food Eng. 2007;78(3):846-54.

[21] Guo C, Zhao L, Yang J, Wang K, Zou J. A novel perspective process for alumina extraction from coal fly ash via potassium pyrosulfate calcination activation method. J Clean Prod. 2020;271:122703.

[22] Birolli WG, da Silva BF, Rodrigues-Filho E. Biodegradation of the fungicide Pyraclostrobin by bacteria from orange cultivation plots. Sci Total Environ [Internet]. 2020;746:140968. Available from: http://www.sciencedirect.com/science/article/pii/S0048969720344971

[23] Yuan J, Li H, Tao W, Han Q, Dong H, Zhang J, et al. An effective method for extracting anthocyanins from blueberry based on freeze-ultrasonic thawing technology. Ultrason Sonochem. 2020;68:105192.

[24] Kahani M, Kalantary F, Soudi MR, Pakdel L, Aghaalizadeh S. Optimization of cost effective culture medium for Sporosarcina pasteurii as biocementing agent using response surface methodology: Up cycling dairy waste and seawater. J Clean Prod. 2020;253:120022.

[25] Agatonovic-Kustrin S, Beresford R. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. J Pharm Biomed Anal. 2000;22:717-27.

[26] Basheer IA, Hajmeer M. Artificial neural networks: Fundamentals, computing, design, and application. J Microbiol Methods. 2000;43:3-31.

[27] Agrawal P, Jayaswal P. Diagnosis and Classifications of Bearing Faults Using Artificial Neural Network and Support Vector Machine. J Inst Eng Ser C [Internet]. 2020 Feb 7;101(1):61-72. Available from: http://link.springer.com/10.1007/s40032-019-00519-9

[28] Zhao J, Ding H, Zhao W, Huang M, Wei D, Jiang Z. Modelling of the hot deformation behaviour of a titanium alloy using constitutive equations and artificial neural network. Comput Mater Sci. 2014;92:47-56.

[29] Anjos O, Iglesias C, Peres F, Martínez J, García Á, Taboada J. Neural networks applied to discriminate botanical origin of honeys. Food Chem. 2015 May;175:128-36.

[30] Martinez-Castillo C, Astray G, Mejuto JC, Simal-Gandara J. Random Forest, Artificial Neural Network, and Support Vector Machine Models for Honey Classification. eFood. 2020;1(1):69-76.

[31] Benali L, Notton G, Fouilloy A, Voyant C, Dizene R. Solar radiation forecasting using artificial neural network and random forest methods: Application to normal beam, horizontal diffuse and global components. Renew Energy. 2019;132:871-84.

[32] Iglesias-Otero MA, Fernández-González M, Rodríguez-Caride D, Astray G, Mejuto JC, Rodríguez-Rajo FJ. A model to forecast the risk periods of Plantago pollen allergy by using the ANN methodology. Aerobiologia (Bologna). 2015;31(2):201-11.

[33] Araujo P, Astray G, Ferrerio-Lage JA, Mejuto JC, Rodriguez-Suarez JA, Soto B. Multilayer perceptron neural network for flow prediction. J Environ Monit. 2011 Jan;13(1):35-41.

[34] Katritzky AR, Pacureanu LM, Slavov SH, Dobchev DA, Karelson M. QSPR study of critical micelle concentrations of nonionic surfactants. Ind Eng Chem Res. 2008;47(23):9687-95.

[35] Fatemi MH, Konuze E, Jalali-Heravi M. Prediction of critical micelle concentration of some anionic and cationic surfactants using an artificial neural network. Asian J Chem. 2007;19(4):2479-89.

[36] Kardanpour Z, Hemmateenejad B, Khayamian T. Wavelet neural network-based QSPR for prediction of critical micelle concentration of Gemini surfactants. Anal Chim Acta. 2005;531(2):285-91.

[37] Belhaj AF, Elraies KA, Alnarabiji MS, Abdul Kareem FA, Shuhli JA, Mahmood SM, et al. Experimental investigation, binary modelling and artificial neural network prediction of surfactant adsorption for enhanced oil recovery application. Chem Eng J. 2021;406:127081.

[38] Khazaei A, Parhizgar H, Dehghani MR. The Prediction of Surface Tension of Ternary Mixtures at Different Temperatures Using Artificial Neural Networks. Iran J Oil Gas Sci Technol. 2014;3(3):47-61.

[39] Gharagheizi F, Eslamimanesh A, Mohammadi AH, Richon D. Use of artificial neural network-group contribution method to determine surface tension of pure compounds. J Chem Eng Data. 2011;56(5):2587-601.

[40] Bakeri G, Delavar M, Lashkenari MS. Surface Tension Prediction of Hydrocarbon Mixtures Using Artificial Neural Network. J Oil, Gas Petrochemical Technol. 2015;2(1):14-26.

[41] Belmadani S, Hanini S, Laidi M, Si-Moussa C, Hamadache M. Artificial Neural Network Models for Prediction of Density and Kinematic Viscosity of Different Systems of Biofuels and Their Blends with Diesel Fuel. Comparative Analysis. Kem u Ind. 2020;69(7-8):355-64.

[42] Rivera RR, Soriano A. Prediction of Density and Speed of Sound of Binary Ionic Liquid and Ketone Mixtures Using Artificial Neural Network. E3S Web Conf. 2019;120:01003.

[43] Safavi A, Sedaghatpour F, Shahbaazi HR. Tensammetric analysis of nonionic surfactant mixtures by artificial neural network. Electroanalysis. 2005;17(12):1112-8.

[44] Jha BK, Tambe SS, Kulkarni BD. Estimating Diffusion Coefficients of a Micellar System Using an Artificial Neural Network. J Colloid Interface Sci. 1995;170(2):392-8.

[45] Samson S, Basri M, Fard Masoumi HR, Abdul Malek E, Abedi Karjiban R. An artificial neural network based analysis of factors controlling particle size in a virgin coconut oil-based nanoemulsion system containing copper peptide. PLoS One. 2016;11(7):e0157737.

[46] Rocabruno-Valdés CI, Ramírez-Verduzco LF, Hernández JA. Artificial neural network models to predict density, dynamic viscosity,

and cetane number of biodiesel. Fuel. 2015;147:9-17.

[47] Asfaram A, Ghaedi M, Goudarzi A, Rajabi M. Response surface methodology approach for optimization of simultaneous dye and metal ion ultrasound-assisted adsorption onto Mn doped Fe3O4-NPs loaded on AC: Kinetic and isothermal studies. Dalt Trans. 2015;44(33):14707-23.

Chapter 6

The Application of Artificial Neural Network to Predicting the Drainage from Waste Rock Storages

Liang Ma, Cheng Huang and Zhong-Sheng Liu

Abstract

Reliable prediction of drainage flow rate and drainage chemistry is essential to the treatment of drainage from waste rock storages at mine sites. The traditional predictive models require simplification and assumption of geo-bio-chemical processes followed by intensive characterization, and sometimes lead to poor prediction accuracy. In the big data era, various sensors are installed in field to constantly monitor mine sites, which enables machine learning to utilize the generated monitoring data and study the underlying pattern behind the data. This chapter describes an approach to use artificial neural network to predict the drainage flow rate and drainage chemistry based on weather monitoring data collected at mine sites. The advantage of this approach is that generally no additional characterization are required to make prediction because the relevant geo-bio-chemical mechanisms are embedded naturally in the monitoring data, which can be captured through machine learning process.

Keywords: machine learning, artificial neural network, drainage flow rate, drainage chemistry, waste rock storages, weather monitoring data

1. Introduction

Hard rock mining generates a huge amount of mine wastes (mine tailings and waste rocks), which often contains metal sulfide minerals. Once exposed to air and water during and after mining, the oxidation of metal sulfides minerals releases acid and heavy metals to the environment. The oxidation of metal sulfides can be accelerated in the presence of microorganisms. The drainage from mine wastes may have high level of toxic elements and chemicals such as arsenic, selenium, lead, uranium, zinc etc. Over time, waste rocks are deposited in the storages which can contain over one hundred million tons and cover a few hundred hectares. The drainage water from waste rock storages and its impact on the surrounding environment are becoming critical challenges to both of mining companies and the public. The treatment of the drainage from waste rock storages may have to last decades, even centuries, and bring a significant cost to the mining sectors [1, 2].

For day-to-day mine site management, flood control and contaminant remediation plan are dependent on the evaluation and prediction of drainage flow rates and drainage chemistries. Various methodologies have been developed over the past several decades to predict the drainage from waste rock storages. For predicting drainage flow rates, a numerical model to simulate groundwater flow through unsaturated bed or layers of earth is developed in [3]. In reference [4], a water balance approach is proposed to calculate the conservation of total water flow through waste rock piles by dividing the whole hydrological process into independent components. In terms of predicting drainage chemistries, a numerous numerical models enabled with mass transport effect are developed to evaluate the geochemical reaction and transport inside waste rock piles [5–7]. Furthermore, using dimensional equation to correlate drainage chemistries with seepage flow rates from waste rock piles is explored in [8]. In reference [9], the effectiveness of a cover system is assessed and a Multiphysics model is developed to predict the iron loading and lime consumption for a full-scale waste rock pile.

However, there are several limitations with above predictive models. For example: 1. Many predictive models are based on lab testing and then scaled up to predict the result in the field, but there is little comprehensive understanding on how to scale up; 2. Simplification and assumptions of geo-bio-chemical processes for the geochemical reaction and leaching process in waste rock storages are critical to the accuracy of the predictive models; 3 Lab or field characterization of material and transport properties related to predictive models is essential, which is also very costly and time-consuming.

To understand and minimize the environmental impact from the contaminated drainage, routine monitoring of waste rock storages is required by many governmental regulators. With the rapid development of computer and sensing technologies, constant and comprehensive monitoring on the waste rock storages is now possible for many mine sites. Daily or even hourly monitoring data become available for many key parameters such as precipitation, temperature, wind, internal temperature, gas concentrations, air/water flow rates, drainage chemistries, etc. These monitoring data are accumulated to weekly, monthly and yearly datasets, and become so huge and complex that traditional data analysis approaches are inadequate to handle and investigate them. As one of the most famous machine learning technologies, artificial neural network not only has the advantages of high processing speed and high computational accuracy, but also enables a machine to mimic human learning behavior and problem solving functions. Thus using neural network to investigate the huge monitoring datasets and further predict drainage flow rates and drainage chemistries from waste rock storages shows very promising potentials. For example, the concentrations of sulphate, chlorine, total dissolved solids and total suspended solids in mine water are predicted by artificial neural network based on the input of pH, temperature and hardness in [10]. Heavy metal included in acid rock drainage is investigated by support vector machine and neural network for a copper mine in Iran [11]. Five machine learning approaches to predict copper concentration are compared in [12]. A feedforward neural network with weather input is proposed to predict drainage flow rates for a full scale waste rock pile [13].

In this book chapter, a refined feedforward neural network based on [13] will be introduced to learn from historical monitoring data and then predict the drainage flow rate, in addition, the refined neural network will also be extended to predict the drainage chemistries in the field. Compared with above traditional predictive models, the proposed neural network approach requires much less simplification and assumption of geo-bio-chemical processes involved and it can significantly reduce characterization cost for mining companies, as the monitoring data inherently contain the information of all the underlying physical mechanisms within real waste rock storages. However, the prediction accuracy is highly dependent on the quality of monitoring data as the proposed neural network is actually a mathematical regression process.

The proposed feedforward neural network selects the weather monitoring data from mine sites as the input to predict and the drainage as the output. The underlying logic for this approach is based on the fact that the water passing through the waste rock storage is mainly from two sources: 1 precipitation falls directly onto the storage and infiltrates into it; 2 groundwater originating from uphill precipitation flows into the storage from higher elevations. Both sources are highly dependent on rain, snow, temperature, hydrologic properties and geo-bio-chemical conditions in the field. As the hydrologic properties and geo-bio-chemical conditions are relatively stable than previous factors related to the weather, the evolution of total precipitation and mean temperature from ambient environment at the mine site is then adopted to correlate with the drainage flow rates and also drainage chemistries. The correlation can be gradually captured by machine learning through studying historical monitoring data from a specific waste rock storage. In addition, the reference [13] proposed to use the number of year and month as additional input to capture long-term fluctuation of drainage. As the number of month is naturally uncycled, the value of the month number has no meaning for machine learning but only brings learning issue when December transits to January. The chapter proposes to use the concept of accumulated days to capture the long term fluctuation instead. With further normalizing all input data, the refined feedforward neural network can better predict the drainage flow rates and further the drainage chemistries. A case study on a full-scale waste rock storage will be provided to validate the proposed approach in this chapter.

2. Methodology

2.1 Feedforward neural network

The Feedforward neural network is an artificial neural network wherein connections between the artificial neurons do not form a cycle, which is different from its variant: recurrent neural networks. The artificial neurons are capable of simulating basic learning behaviors through receiving inputs, calculating a weighted sum and then passing the sum through a transformation known as activation function to produce outputs. The mathematical calculation for an artificial neuron in a feedforward neural network is generally illustrated as follows:

$$\boldsymbol{X}_{t,p} = \varphi_{t,p}\left(\sum_{i=1}^{q} \boldsymbol{X}_{t-1,i}\boldsymbol{w}_{t,p,i} + \boldsymbol{b}_{t,p}\right) \tag{1}$$

where $\boldsymbol{t}$ denotes the layer number and $\boldsymbol{p}$ denotes the order number in that layer. The combination of $\boldsymbol{t}$ and $\boldsymbol{p}$ can be used to determine the location of the neuron in the network. For the $\boldsymbol{p}$th neuron located at the $\boldsymbol{t}$th layer, there are q inputs from $\boldsymbol{X}_{t-1,1}$ through $\boldsymbol{X}_{t-1,q}$, and also weights from $\boldsymbol{w}_{t,p,1}$ to $\boldsymbol{w}_{t,p,q}$. In addition, $\boldsymbol{b}_{t,p}$ is the bias input, $\varphi_{t,p}$ is the activation function and $\boldsymbol{X}_{t,p}$ is the output of that neuron. After calculating the output of $\boldsymbol{X}_{t,p}$ based on Eq. (1), it may further propagate to the input of the next layer or leave the network as the output of the whole feedforward neural network.

2.2 Application to predict the drainage from waste rock storage

The schematic of the proposed feedforward neural network structure is illustrated in **Figure 1**. As mentioned above, hydrologic properties and geo-bio-chemical conditions in waste rock storages are generally considered as a very slow evolution, which means they are relatively stable compared with weather conditions such as rain, snow and temperature in the field. The dynamics of weather conditions are powerful to act as driving input forces for the training process, leaving hydrologic properties and geo-bio-chemical conditions as coefficients within neural network to be determined during learning process. As the temperature controls not only the formation of rain/snow but also evaporation rate on the surface, total precipitation and mean temperature are then selected as two groups of neurons in the input layer. Current and preceding total precipitation and mean temperature are extracted from the weather monitoring database, then they are formatted into a time series in the input layer before entering the hidden layer. The length of time series determines the neuron number in each group of the input layer. For example, an input including previous 10 daily weather monitoring data indicates 10 neurons for previous daily total precipitation and 10 neurons for previous daily mean temperature in the input layer.

An additional neuron in the input layer is composed of a time tag that represents the drainage measurement day. For example, the day with first weather monitoring data is considered as the first day, and the corresponding time tag is set to 1. Any future time tag for one drainage measurement is the accumulated day number adding from the first day to the measurement day. By introducing the concept of accumulated day number as the time tag, the geo-bio-chemical evolutions inside the waste rock storages no longer have to keep constant in the temporal scale, and become potentially time dependent. Thus this hybrid input structure enables the proposed neural network to capture the long-term trend of the drainage flow rates and drainage chemistries.

The output can be calculated by moving forward in the neural network based on Eq. (1). After the output is obtained, it is compared with drainage monitoring data (target) including flow rate and chemistry concentration. A cost function is then adopted to evaluate the difference between output and target. In this study, the mean squared errors (*MSE*) is used as the cost function, which is the average squared difference between calculated outputs and the target. The calculation of *MSE* is as follows,

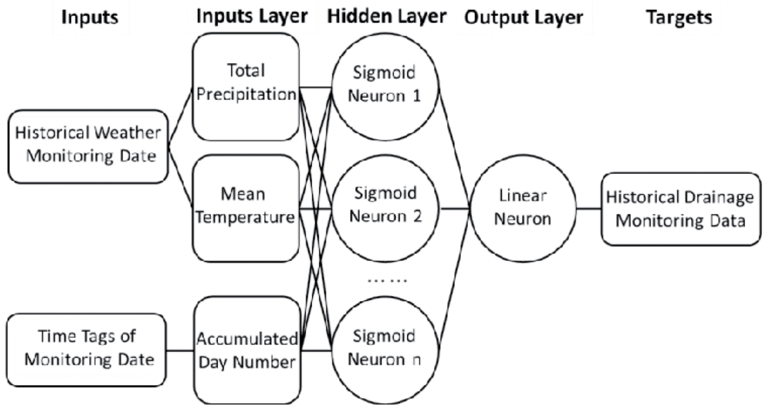


Figure 1.
The proposed feedforward neural network structure.

$$MSE = \frac{1}{m}\sum_{i=1}^{m}\left(o_i - t_i\right)^2 \tag{2}$$

where o_i is the i th calculated output, t_i is the i th target, and m is the number of the target for machine learning.

As both of input and target data are different in terms of their scales, it is generally required to pre-process them to become normalized before the training starts. The normalization could accelerate the training process by making all undetermined coefficients in the neural network get updated in the same scales. For this study, the mean for each data set (total precipitation, mean temperature, flow rate, chemistry concentration) is set to 0 and the standard deviation is set to 1. The normalization is obtained by following calculations:

$$\bar{x} = \sum_{i=1}^{n} x_i / n \tag{3}$$

$$s = \sqrt{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)/n} \tag{4}$$

$$y_i = \left(x_i - \bar{x}\right) \oslash s \tag{5}$$

where $\bar{x}$ and s is the mean and standard deviation for the data set of x . n is the total number of data in the set, $\oslash$ denotes Hadamard division, and y is the normalization of x .

In the beginning, all coefficients within the neural network shown in Eq. (1) are randomly initiated. During the training process, they are automatically updated through a special data iteration technique called backpropagation algorithm, which calculates the gradient of the cost function based on comparing target with output. The proposed feedforward neural network should be trained with a fair amount of observation samples from historical monitoring database so that it can capture the correlation between input data and target data. Here an observation sample is defined as a combination of input and target from historical monitoring database. The training needs assessment to prevent both of underfitting and overfitting with various validation methods. The hold-out approach is adopted in this study. Among all observation samples, the training observation samples are those for actual training, the validation observation samples are for evaluating the generalization of the neural network and the training process continues until the generalization does not get improved, and the rest are called testing observation samples which do not impact on the training process but give independent assessment for the training performance.

After the training is completed, both of MSE and R are used to estimate the training performance. Here R value measures the correlation between calculated results (output) and measured ones (target), which is calculated as follows,

$$R = \frac{m(\sum o \times t) - (\sum o)(\sum t)}{\sqrt{\left[m\sum o^2 - (\sum o)^2\right]\left[m\sum t^2 - (\sum t)^2\right]}} \tag{6}$$

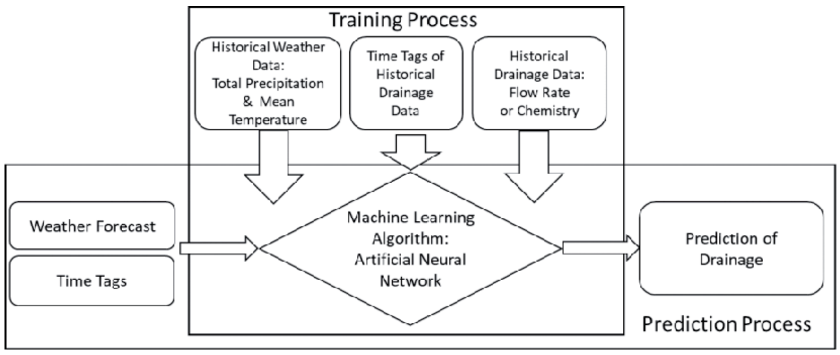

Training Process
Historical Weather Data: Total Precipitation & Mean Temperature
Time Tags of Historical Drainage Data
Historical Drainage Data: Flow Rate or Chemistry
Weather Forecast
Time Tags
Machine Learning Algorithm: Artificial Neural Network
Prediction of Drainage
Prediction Process

Figure 2.
The functions of the proposed feedforward neural network.

Theoretically, the lower value for ***MSE*** and the closer to 1 for ***R***, the better training performance.

In theory, a well-trained neural network proposed in this study is able to reasonably predict future drainage flow rate and drainage chemistry concentration for full-scale waste rock storages as long as the historical weather monitoring database, historical drainage monitoring database and the weather forecast onsite are available. **Figure 2** shows the schematic diagram of the general functions for the proposed feedforward neural network approach. There are two processes involved in the implementation of the approach. After the training processed is completed, the correlation between weather and drainage for the waste rock storage is believed to be captured by the proposed neural network, and then the prediction process starts to utilize weather forecast to predict future drainage on site.

3. Validation-a case study

3.1 Input, target and neural network parameters

To validate the proposed neural network approach, a full-scale case study is performed to predict real drainage flow rate and drainage chemistry in field condition. A real waste rock pile from an anonymous mine site in western Canada is adopted in this study. The proposed neural network is trained by historical monitoring data for 16 years (Year 1 to Year 16). After the training is completed, it will be used to predict the drainage in the next 2 years (Year 17 and Year 18). A comparison between real measurement and predicted value will be provided.

At this mine site, the drainage flow rates are not directly measured but they are calculated based on readings of the water level in v-notch weirs installed at the end of the drainage collecting ditch. Thus the v-notch is used as the one actual target of neural network training. In the following discussion, the drainage flow rate actually refers to the original measurement of v-notch from the weirs. Among all drainage chemistry data from this waste rock pile, acidity is selected for this case study as another target, because it is directly linked to the lime consumption for contaminant treatment. These drainage measurement are generally performed in a dynamic time frame at this mine site. During spring freshet and large precipitation periods, the measurements are usually more frequent than the remaining time in a year, as increased drainage flow rate is observed.

The weather monitoring data at this site is extracted from the website of Environment Canada (weather.gc.ca) on a daily basis, including minimum temperature, maximum temperature, mean temperature, total rain, total snow, total precipitation and snow thickness on ground, etc. For this case study, the total 16 years weather monitoring data have been obtained for the training purpose. As mentioned in the previous section, two independent weather parameters measured on a daily basis - the total precipitation and mean temperature are selected as the inputs for the proposed neural network.

To evaluate how long the weather can impact on the drainage from this waste rock pile, two types of input layers are proposed for comparison in the study. When a target (flow rate or acidity) is selected to train the neural network, its measurement date is extracted. Daily average input layer consists of 21 neurons including the time tag of measurement day, daily total precipitation and daily mean temperature from previous 9 days and the measurement day, which mainly investigates short-term weather impact on the drainage. Furthermore, weekly average input layer has 21 neurons including the time tag of measurement day, weekly average total precipitation and weekly averaged mean temperature from previous 9 weeks and current week to investigate long-term weather impact. The weekly average input layer reflects longer weather monitoring data than the daily average input layer does, however, high frequent information is filtered in the weekly average input layer. The summary of daily average and weekly average input layers can be found in **Table 1**. Here 0 day means the measurement day, 0 week means measurement day and previous 6 days. Finally, both types of input layers are adopted to train the neural network to determine which input layer is more competent to capture the underlying pattern and make a better prediction.

As the mine site is anonymous, the original monitoring data is confidential and not publicly accessed. To protect the site information, only normalized historical monitoring data including total precipitation, mean temperature, flow rate and acidity from the 16 years are provided in **Figure 3**. Total numbers of flow rate measurement and acidity measurement during the 16 years is 1741 and 320, respectively. It should be noticed that the weather data are extracted on a daily basis and any missing data is represented by a gap. The flow rate and acidity is not measured in a fixed time frame and the time interval is dynamic, so each measurement data is represented by a solid dot in the figure. As some weather data are missing, not all of drainage measurements are utilized for the training. Those drainage measurements that do not have a complete daily average or weekly average input will be excluded. In terms of the hold-out approach to avoid overfitting, 80% of the total observation samples are used for training and 20% for validation. No testing data is allocated

Type of input layer	Daily Average	Weekly Average
Neuron number	21	21
Description	Mean temperature of −9 day Total precipitation of −9 day Mean temperature of −8 day Total precipitation of −8 day … … … … Mean temperature of −1 day Total precipitation of −1 day Mean temperature of 0 day Total precipitation of 0 day Time tag of measurement day	Mean temperature of −9 week Total precipitation of −9 week Mean temperature of −8 week Total precipitation of −8 week … … … … Mean temperature of −1 week Total precipitation of −1 week Mean temperature of 0 week Total precipitation of 0 week Time tag of measurement day

Table 1.
Description of daily average and weekly average input layers.

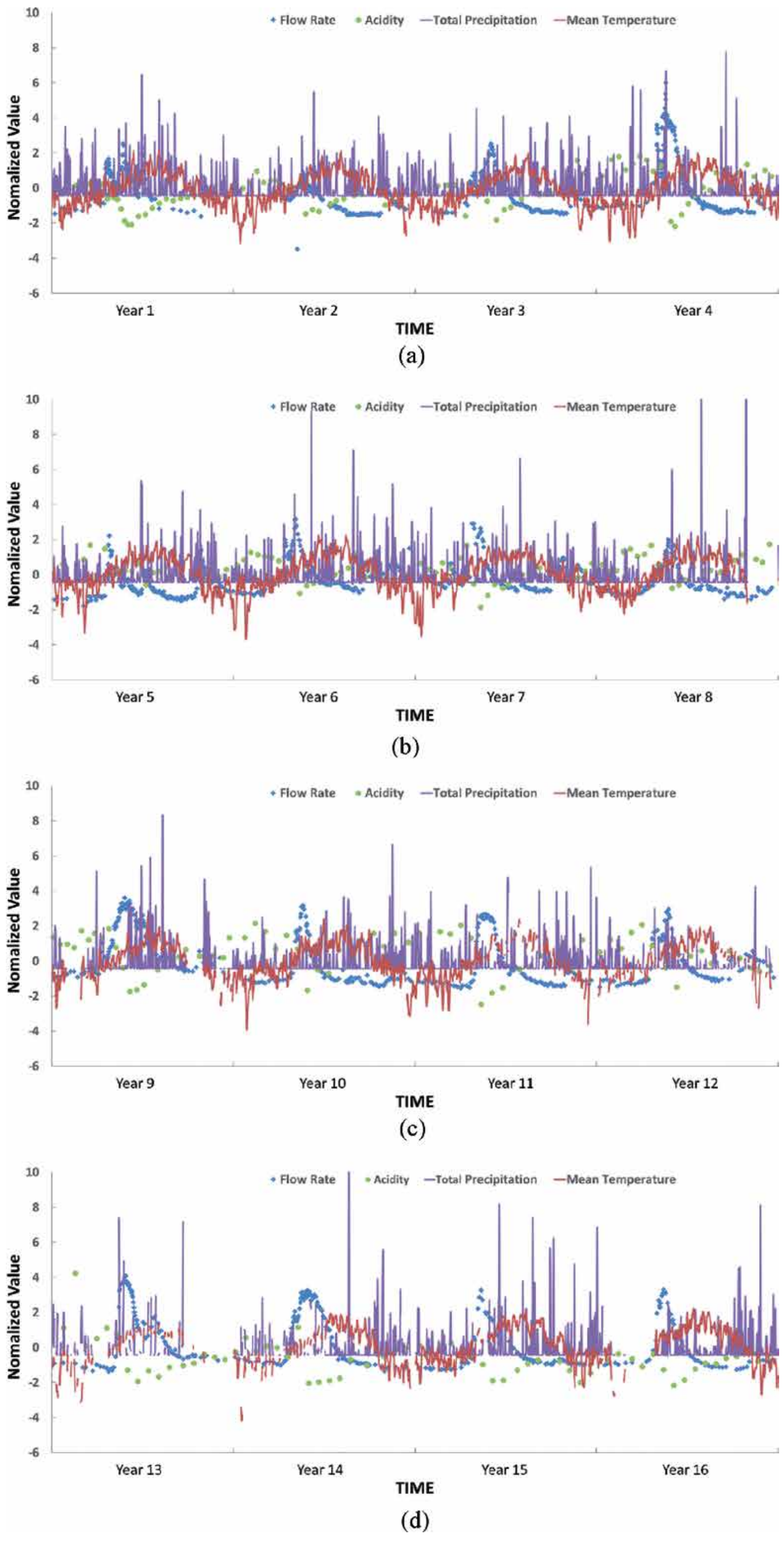


Figure 3.
Normalized monitoring data for the 16 years (year 1 to year 16). (a) Year 1 to year 4; (b) year 5 to year 8; (c) year 8 to year 12 and (d) year 13 to year 16.

in the training process, as a prediction of the drainage in the next 2 years will be performed after the training is completed. As the monitoring data of weather and drainage during the next 2 years are also available. The real weather data will be utilized as the input of the neural network. In real situation, weather forecasting data should be adopted for prediction purpose. The predicted drainage flow rates and acidities will be compared with the real monitoring values.

As shown in the **Figure 1**, only a single hidden layer is in the feedforward neural network adopted for this case study. So the number of neurons in the hidden layer is considered as a hyperparameter for the training process. A grid search is performed on 5, 10 and 20 neurons in the hidden layer to find the optimized size. The proposed neural network is trained through Levenberg–Marquardt backpropagation algorithm. The adaptive value (damping factor) is set to 0.001 initially, and will increase by 10 until the change of above results in a reduced performance value. The change is then made to the network and adaptive value is decreased by 0.1. The maximum adaptive value is set to 1e10. The maximum epochs before the stop of training is set to 1000. However, the training may be stopped early if the ***MSE*** on the validation vectors stops to improve or remains the same for 6 epochs in a row.

The proposed neural network is implemented through the machine learning toolbox built in commercial software MATLAB. The weather monitoring data are pre-constructed for both types of input layer through Microsoft Excel Macro before exporting into the MATLAB.

3.2 Regression and prediction results

After the training is completed, all training and validation observation samples go through Eq. (1) again and then the calculated output is called drainage regression. The ***MSE*** and ***R*** between drainage regression and real measured ones (targets) are listed in **Table 2** based on a grid search is performed on both of daily average input layer and weekly average input layer with 5, 10 and 20 neurons in the hidden layer. It is observed that results from the neuron network with weekly average input layer are generally better than those obtained from the neuron network with daily average input layer, indicating that both of flow rate and acidity from this waste rock pile are mainly controlled by long-term weather trend rather than short-term one. For flow rate regression, the lowest *MSE* is obtained from both of 10 and 20 neurons in hidden layer, so the neuron network trained by weekly average input layer with 10 neurons in hidden layer is selected as the successful candidate to predict future drainage flow rates as it has smaller size and less undermined coefficients than the network with 20 neurons in the hidden layer. For acidity regression, the lowest ***MSE*** is obtained from the neuron network with weekly average input layer and 20 neurons in the hidden layer, which is selected to predict future drainage acidities.

Number of Neurons in the Hidden layer	**Flow Rate (*MSE, R*)**		**Acidity (*MSE, R*)**	
	Daily Average	**Weekly Average**	**Daily Average**	**Weekly Average**
5	1.01, 0.62	0.25, 0.93	0.53, 0.65	0.35, 0.78
10	0.99, 0.63	0.18, 0.95	0.55, 0.65	0.41, 0.79
20	0.92, 0.67	0.18, 0.95	0.58, 0.67	0.28, 0.84

Table 2.
Results of the grid search.

The comparison between regressions of flow rate and acidity and real measurements (target) in temporal scale are provided in **Figure 4**. It is found that the proposed neural network is capable to capture the underlying correlation between the drainage and weather, as not only seasonal fluctuations in a year but also long term evolution across years are well reflected in the drainage regression.

In addition, the successful candidate neuron networks (weekly average input layer with 10 neurons in hidden layer for flow rate, and weekly average input layer with 20 neurons in hidden layer for acidity) are adopted to predict the future flow rate and acidity in the Year 17 and Year 18 based on the input extracted from the real weather monitoring data. The prediction and real measurement in temporal scale are compared in **Figure 5**.

It is observed that the general trend for both of flow rate and acidity are well predicted by the neural network. The proposed neural network is capable to predict the time and also the amount of peak flow rates in the spring freshet of both years, which is important for site water management. In terms of acidity, the long term downward trend shown in Year 11 to Year 16 is reflected in the prediction, which matches the trend of real measurement in both years. However, seasonal fluctuation has some mismatch. The reason is that the amount of observation samples for acidity is much less than those for flow rate, so the acidity prediction is not as good as the flow rate prediction in this case study. The accuracy can be improved when more monitoring data are accumulated for the training in the future.

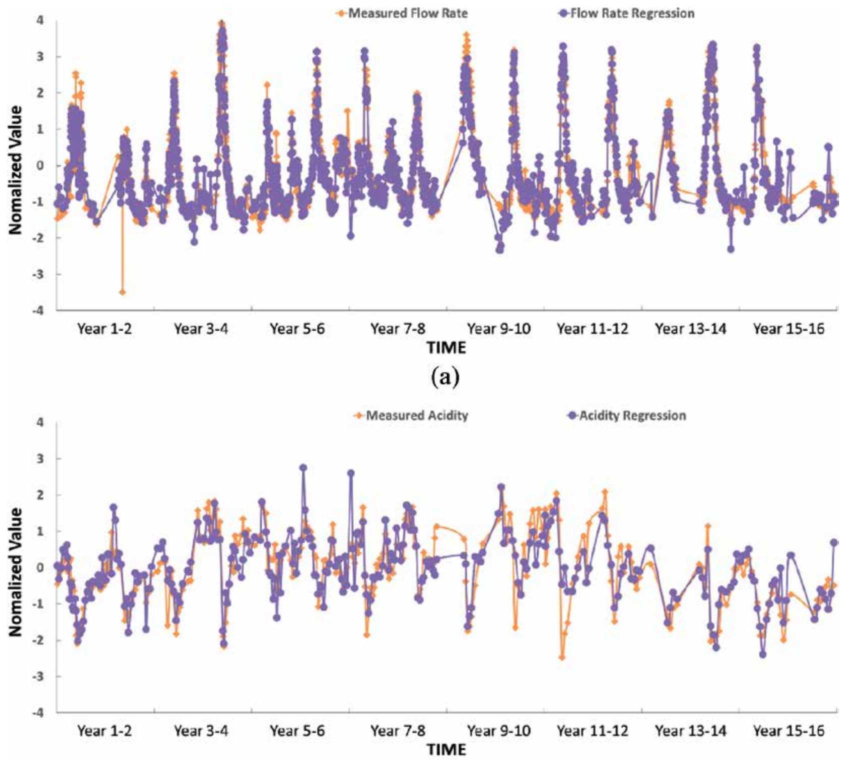


Figure 4.
Drainage regression vs. real measurement for the 16 years (year 1 to year 16). (a) Flow rate and (b) acidity.

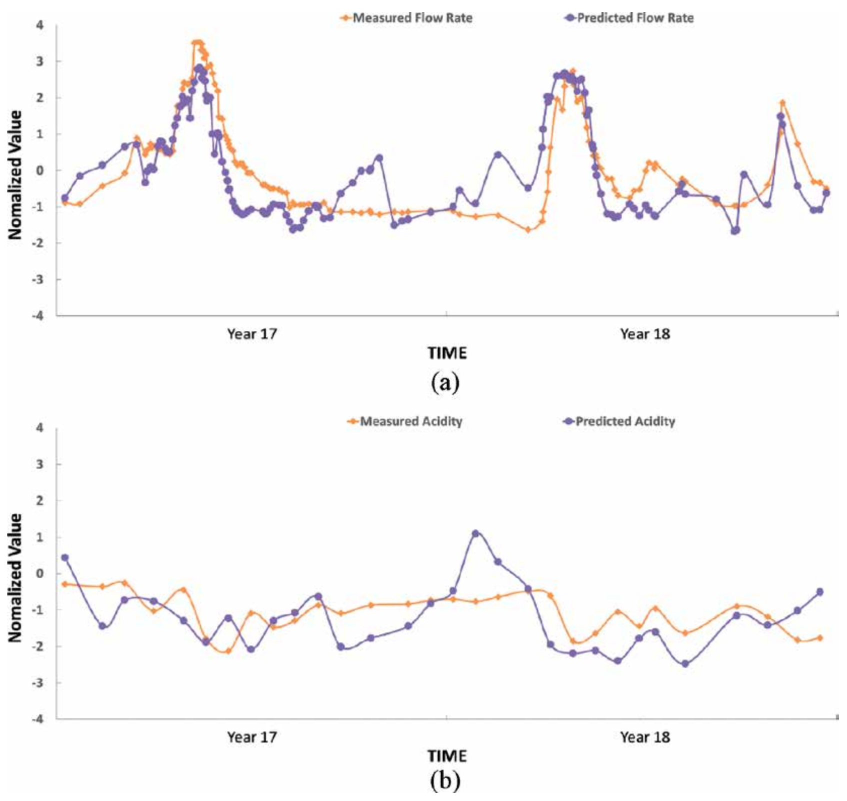


Figure 5.
Drainage prediction vs. real measurement for the next 2 years (year 17 and year 18). (a) Flow rate and (b) acidity.

4. Conclusions

A machine learning algorithm based on feedforward neural network is introduced in this chapter to correlate the drainage flow rate and drainage chemistry with the field precipitation and temperature for waste rock storages. Comparing with traditional predictive models, the neural network approach requires little simplification and assumptions of bio-geo-chemical processes involved, in additional, the cost and time for characterizations can be significantly reduced. The advantage of the neural network is that all underlying mechanisms have been naturally reflected in the monitoring data, which can be gradually captured during the machine learning process.

A case study on a full-scale waste rock storage is performed. The results show that the flow rate and acidity of the drainage discharged in the field have strong correlations with previous 10 weekly averaged weather data at this site. The capability of making prediction of future drainage in the field is also validated. However, the structure of input layer, hidden layer number, neurons in the hidden layer are all site specific, which may be adjusted for the applications to other waste rock storages.

It should also be addressed that the measurements of drainage flow rate and drainage chemistry may not always be accurate in the field, furthermore, they can fluctuate in a single day depending on the hydrogeological conditions. So the monitoring data may not represent the daily average in some cases, which means that some mismatch between the prediction and measurement does not necessarily indicate the prediction is wrong. High frequent (multiple in a single day or hourly) measurement is highly suggested, so that good quality monitoring data will be available to predict drainage for waste rock storages in the future.

Acknowledgements

The work is financially supported by the Environmental Advances in Mining Program and New Beginnings Initiative in the National Research Council Canada.

Author details

Liang Ma, Cheng Huang* and Zhong-Sheng Liu
National Research Council Canada, Vancouver, Canada

*Address all correspondence to: cheng.huang@nrc-cnrc.gc.ca

References

[1] Policy for Metal Leaching and Acid Rock Drainage at Minesites in British Columbia, July 1998.

[2] Hudson-Edwards K: Tackling mine wastes. Science. 2016; 352(6283): 288-290.

[3] Lahmira B, Lefebvre R, Aubertin M, Bussiere B. Effect of material variability and compacted layers on transfer processes in heterogeneous waste rock piles. Journal of Contaminant Hydrology. 2017; 204: 66-78.

[4] Isabel D, Gelinas PJ, Bourque E, Nastev M, Precourt S. Water budget for the waste rock dump at La Mine Doyon, Quebec. 1994; MEND report 1.14.2d.

[5] Lefebvre R, Hockley D, Smolensky J, Gelinas P. Multiphase transfer processes in waste rock piles producing acid mine drainage 1: Conceptual model and system characterization. Journal of Contaminant Hydrology. 2001; 52: 137-164.

[6] Molson JW, Fala O, Aubertin M, Bussiere B. Numerical simulations of pyrite oxidation and acid mine drainage in unsaturated waste rock piles. Journal of Contaminant Hydrology. 2005; 78: 343-371.

[7] Ma L, Huang C, Liu ZS, Morin K, Aziz M, Meints C. Prediction of acid rock drainage in waste rock piles Part 1: Water film model for geochemical reactions and application to a full-scale case study. Journal of Contaminant Hydrology. 2019; 220: 98-107.

[8] Liu ZS, Huang C, Ma L, Dy E, Xie Z, Tufa K, et al. The characteristic properties of waste rock piles in terms of metal leaching. Journal of Contaminant Hydrology. 2019; 226: 103540.

[9] Ma L, Huang C, Liu ZS, Morin KA, Dy E, Tufa K, Fisher E, Zhou J, Aziz M, Meints C. A Full-Scale Case Study on the Leaching Process of Acid Rock Drainage in Waste Rock Piles and the Net Infiltration through Cover Systems. Water, Air, & Soil Pollution. 2020; 231(6): 1-16.

[10] Khandelwal M, Singh T. Prediction of mine water quality by physical parameters. Journal of Scientific & Industrial Research. 2005; 64: 564-570.

[11] Aryafar A, Gholami R, Rooki R, Ardejani FD. Heavy metal pollution assessment using support vector machine in the Shur River, Sarcheshmeh copper mine, Iran. Environmental earth sciences. 2012; 67(4): 1191-1199.

[12] Betrie GD, Tesfamariam S, Morin KA, Sadiq R. Predicting copper concentrations in acid mine drainage: a comparative analysis of five machine learning techniques. Environmental monitoring and assessment. 2013; 185(5): 4171-4182.

[13] Ma L, Huang C, Liu ZS, Morin KA, Aziz M, Meints C. Artificial Neural Network for Prediction of Full-Scale Seepage Flow Rate at the Equity Silver Mine. Water, Air, & Soil Pollution. 2020; 231:1-15.

Chapter 7

Risk Assessment and Automated Anomaly Detection Using a Deep Learning Architecture

Stelios C.A. Thomopoulos

Abstract

Risk-based security is a concept introduced in order to provide security checks without inconveniencing travelers that are being checked with unqualified scrutiny checks while maintaining the same level of security with current check point practices without compromising security standards. Furthermore, risk-based security, as a means of improving travelers' experience at check points is expected to reduce queueing and waiting times while improving at the same travelers' experience during checks. A number of projects have been funded by the European Commission to investigate the concept of risk-based security and develop the means and technology required to implement it. The author is the Coordinator of two of the flagship projects funded by EC on risk-based security: FLYSEC and TRESSPASS. This chapter discusses and analyses the concept of risk-based security, the inherent competing mechanism between risk assessment, screening time and level of security, and means to implement risk-based security based on anomaly detection using deep learning and artificial intelligence (AI) methods.

Keywords: risk assessment, security, anomaly detection, deep learning, neural networks, crowd simulation, control and command, surveillance, risk-based security

1. Introduction

Risk-based security is built around the premise that information obtained from observable aspects of human identity and possession and knowledge acquired about hidden aspects of human capability and intent can be intelligently combined to assess to some great extent of accuracy the threat a given individual poses to a security system, be it an airport or a border crossing point (BCP). Then, in turn, associating the estimated level of threat with a measure of risk by factoring in the cost that the assessed threat can represent to the system that is being secured by taking into account the impact and cost a given threat can have on a security system, a risk-based security approach can be designed and implemented, whereby security checks are tailored to be commensurate to the estimated risk each individual may pose, instead of being uniform irrespectively of the risk posed by each individual, as is the case today. Taking into account that less than 5% of all individuals can be a potential security risk, the savings in terms of time required to go through risk-based security systems with speedier tests for the 95% of low to no risk individuals can be significant, waiting times in security lines can be reduced and thus the level of comfort and customer satisfaction be drastically improved.

The concept of *risk-based security* is founded on the premise that less than 5% of travelers represent a threat to the security of a border crossing point (BCP), it is conceivable that by somehow identifying the risk-free travelers, the security checks for those "trusted" travelers can be relaxed and sped up, leading into lower delays in the security screening systems. By easing off the security checks on the "trusted" 95% of travelers, the security screening process can focus on the potentially "suspicious" 5% of travelers, thus increasing the odds of identifying them more efficiently.

The concept of risk-based security is indeed promising in terms of improving travelers' experience by easing off security screening and reducing the overall time required to spend at a security check-point. However, the difficulty in implementing a risk-based security systems lies on: (a) developing and implementing non-intrusive, GDPR[1] compliant technology and systems that can estimate the risk level of each traveler without inducing additional and cumulative delays; (b) testing such systems before rolling them out in operational environments; and (c) estimate their performance and efficacy under ideal conditions for obtaining performance bounds, calculating the cost of the required investment for implementing risk-based technologies; and (d) calculate the degradation in performance when moving away from the "ideal" operational conditions into realistic operational conditions.

The European Union (EU) and other international organizations promote this approach through various initiatives. The European Commission (EC) issued the "Smart Borders package" which aims to modernize the Schengen area's external border management by improving the quality and efficiency of border crossing processes through the establishment of 'Stronger and Smarter Information Systems for Borders and Security' [1]. The International Air Transport Association (IATA) proposed a Checkpoint of the Future, designed to enhance security while reducing queues and intrusive searches at airports by using intelligence-driven risk-based measures [2]. Along these lines the EC funded the Research and Innovation project FLYSEC [3] has developed and demonstrated an innovative, integrated, and end-to-end airport security system facilitating risk-based screening with the introduction of novel intelligent technologies.

This chapter *discusses* a model of risk-based security developed over a number of EU funded projects, *highlights* the need to using simulation in assessing the efficacy of risk-based security technologies and protocols, and *elaborates* on the use of AI and deep learning algorithms for assessing the perceived risk for each traveler based on observable behavioral indicators (parameters), while *factoring in* information acquired from various sources about hidden behavioral parameters.

2. Conventional versus risk-based security

Figure 1 shows a today's conventional security check point whereby we distinguish two types of checks: (a) the "normal" check where all individuals in the security check area are treated uniformly by applying the same level of security for all; and (b) the "increased" security check point where travelers are channeled if they fail at normal security check point. It should be pointed out that in this security system of check points, currently implemented almost worldwide, the "increase inspection" is usually the outcome of randomized selection of travelers to be subjected to an increased level of inspection and is usually based on the principle of "importance sampling[2]" methods. These methods try to detect a probabilistic event, such as the existence of a suspect among travelers, with a certain degree of confidence by taking

[1] GDPR compliance: Complete guide to GDPR compliance: https://gdpr.eu/

[2] Importance sampling definition: https://en.wikipedia.org/wiki/Importance_sampling

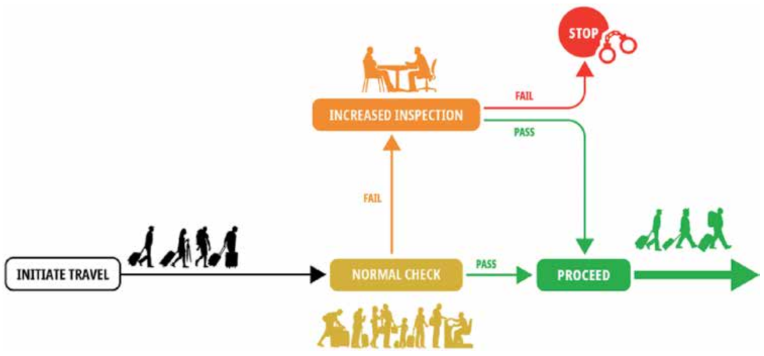


Figure 1.
Today's security check-point concept (curtesy of TRESSPASS).

into account the probability of existence of such an event and possibly the range of values the event can assume. These methods are "blind," that it they draw samples from the distribution indiscriminately and without takin into account any specific attributes of the samples, and thus, they are also GDPR compliant. As it will be pointed out further down in the chapter, risk-based methods need to pay special attention to comply with GDPR as they gather and use information and knowledge about individuals' private data such as identity, possession, capability, and intent.

Risk-based security associates the estimated risk for each traveler with a commensurate level of security scrutiny. Using prior information about each traveler and sensory data obtained while the traveler is within the security perimeter of a monitored area, a risk-based security system assigns a risk factor to each traveler and depending on the value of the risk factor, the traveler is mapped to a level of security scrutiny commensurate with the perceived risk. Although different number of levels can be associated with the estimated risk, for practical reasons, it is sufficient to associate the entire range of risk values into three different levels of security, Trusted/Registered (Green), Casual (Yellow) and Enhanced Security (Red), as shown in **Figure 2** [5].

In **Figure 2**,[3,4] a number of GDPR-compliant technologies that can be used for and contribute to the risk assessment are shown in and include: mobile app way

[3] FLYSEC … . Complementing the ACI/IATA efforts, the FLYSEC European H2020 Research and Innovation project (http://www.fly-sec.eu/) has developed and demonstrated an innovative, integrated and end-to-end airport security process for travelers, enabling a guided and streamlined procedure from the landside to airside and into the boarding gates, and offering for an operationally validated innovative concept for end-to-end aviation security. FLYSEC has contributed towards: (i) innovative processes facilitating risk-based screening; (ii) deployment and integration of new technologies and repurposing existing solutions towards a risk-based Security paradigm shift; (iii) improvement of traveler facilitation and customer service, bringing security as a real service in the airport of tomorrow; (iv) achievement of measurable throughput improvement and a whole new level of Quality of Service; and (v) validation of technologies, devices and applications that can be used to assess risk while the travelers move around in the security perimeter.

[4] TRESSPASS … TRESSPASS focusses on risks emerging from people that use the BCP such as travelers and staff, including people that act as such. This includes their luggage, both checked in and hand-held. A typical DBT for a BCP is based on a subset of a typical set of attributes regarding such persons and their travel group. In a DBT, threats are described using as building blocks in terms of ***Observable aspects***: (a) *identity*: specific people of which we know that they cause, or will not cause, a threat; and (b) *possession*: assets that we know that can be used to generate a threat, e.g. explosives; whereas in terms of ***Hidden aspects***: (c) *capability*: people with specific skills with which they can, generate a threat; and (d) *intent*: people that have an intent from which a threat can be derived.

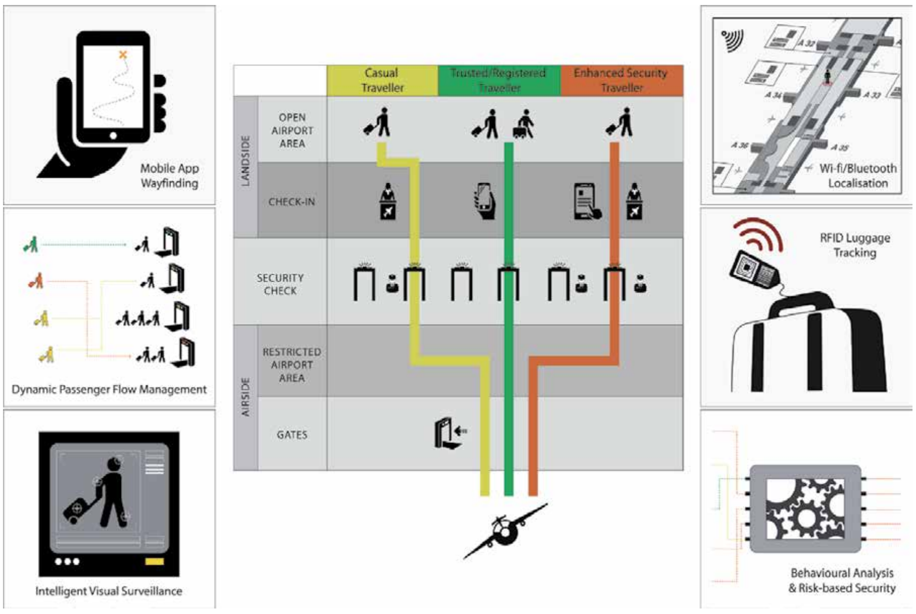

Mobile App Wayfinding
Dynamic Passenger Flow Management
Intelligent Visual Surveillance
Casual Traveller
Trusted/Registered Traveller
Enhanced Security Traveller
LANDSIDE
OPEN AIRPORT AREA
CHECK-IN
SECURITY CHECK
AIRSIDE
RESTRICTED AIRPORT AREA
GATES
Wi-fi/Bluetooth Localisation
RFID Luggage Tracking
Behavioural Analysis & Risk-based Security

Figure 2.
Association of three security scrutiny levels, namely "enhanced security," "casual traveler," and "trusted/registered" with the estimated level of risk for each traveler. These three levels have been introduced in the FLYSEC project [3] and carried over to the TRESSPASS project [4].

finding; dynamic travelers flow management; intelligent visual surveillance; Wi-Fi/Bluetooth localization; RFID mobile tracking; and behavioral analysis & risk-based security personnel mobile app.

Figure 3 represents a risk-based security check point that results from combining the three-level risk-based security screening of **Figure 2** with the conventional security screening of **Figure 1**. As it can been seen from **Figure 3**, the need for assessing each traveler's risk factor from various observable parameters requires measuring somehow these parameters, of course in a GDPR compliant way, and thus additional processing steps and capabilities that may induce additional delays in screening process. Thus, the fundamental premise of risk-based security as a

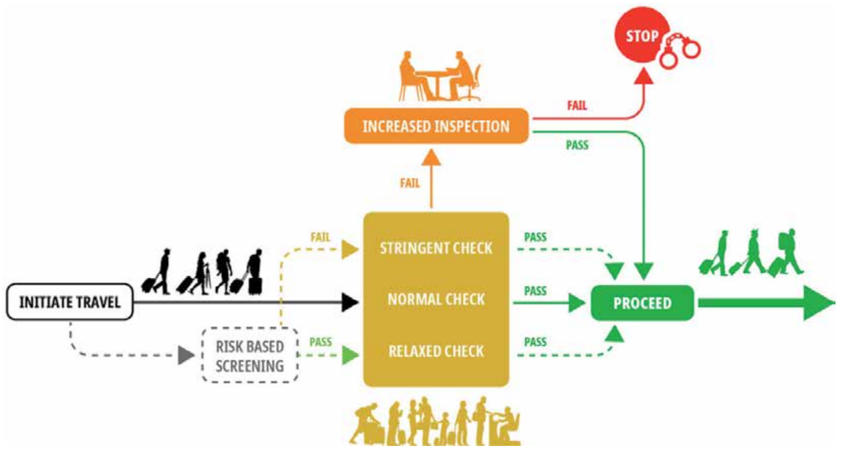


Figure 3.
Risk-based security check point: The standard (randomized scrutiny checks) security check point of ***Figure 2****, has been modified by introducing a three-level risk assessment process prior to the security scrutiny resulting in three different security scrutiny levels at the security screening check point (reference).*

means of providing the same, at least, level of security as conventional check points without inducing additional delays, seems to be in conflict with the additional delay induced by additional screening tests required for estimating each traveler' risk index, unless the risk assessment process is done transparently while the travelers move from the entry to exit points in a BCP (Border Crossing Point).

Figures 4 and **5** depict two block diagrams implementing the conventional security screening process of **Figure 1** and the risk-based security screening process of **Figure 3** respectively. From the two diagrams it is clear that additional screening stages are required for assessing the risk for each traveler in risk-based security. Each one of these additional risk assessing stages induce additional delays in the security screening process, that add up to an overall additional time required for risk-based security screening compared to the time required for security screening through a conventional security check point.

Thus, it appears that risk-based security may require additional processing time for estimating risk that may offset the benefits from faster security screening for those travelers whose estimated risk classifies them in either the "trusted/registered traveler" or "casual travelers" categories for whom security screening is relaxed and thus faster than the time would be required to screen them in today's conventional

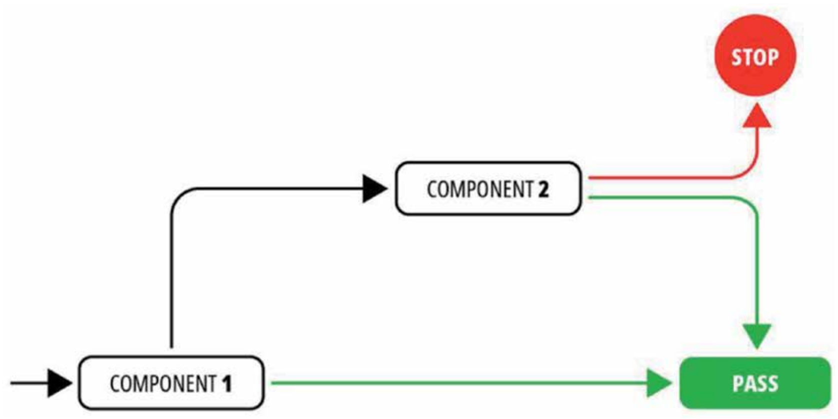


Figure 4.
Configuration 1 (current BCP implementation).

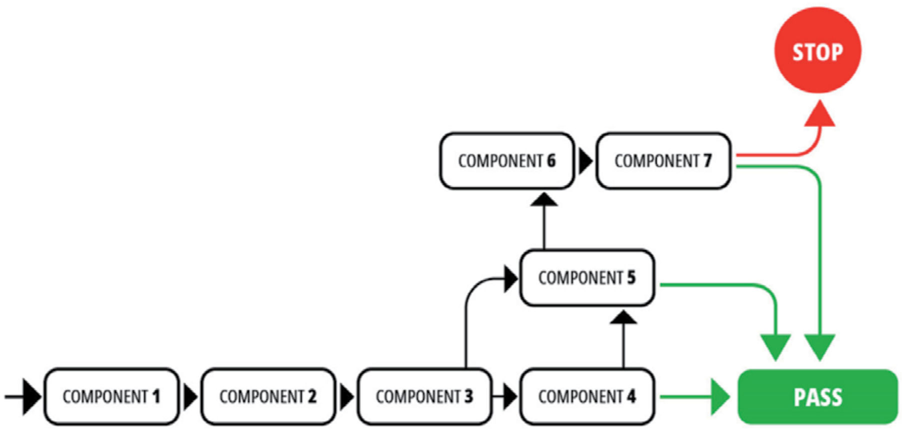


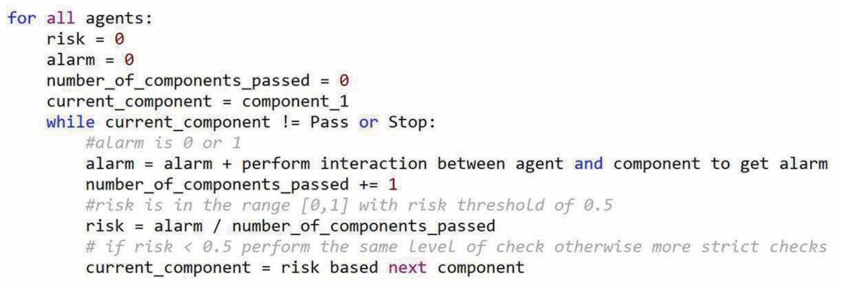
Figure 5.
Configuration 2 (risk-based BCP implementation).

check points of **Figure 1**. Granted that over 90% of travelers fall within these two categories and will experience reduced delays at security screening, it remains to determine if the aggregate benefits from the reduced security screening at check points will trade off positively against the additional delays induced by the additional screening points for determining each traveler's risk as in **Figure 5**.

In order to quantify the cost–benefit trade-offs between the efficiency of a risk-based security BCP and the delay induced by additional checks required for assessing risk, the following experiment was conducted using Fraunhofer's FhG BCP Monte-Carlo agent-based simulator of a BCP configuration (curtesy of Fraunhofer Institute) [4].

For the simulation, we assumed a BCP with 1000 travelers, some exhibiting normal (no risky) behavior, whereas the rest exhibit suspicious behavior, with the following parameters:

- Distribution of traveler types: [Normal, Suspicious]: [0.9,0.1]
- Alarm threshold for each component: 0.5
- Risk calculation: According to the script below:



```
for all agents:
    risk = 0
    alarm = 0
    number_of_components_passed = 0
    current_component = component_1
    while current_component != Pass or Stop:
        #alarm is 0 or 1
        alarm = alarm + perform interaction between agent and component to get alarm
        number_of_components_passed += 1
        #risk is in the range [0,1] with risk threshold of 0.5
        risk = alarm / number_of_components_passed
        # if risk < 0.5 perform the same level of check otherwise more strict checks
        current_component = risk based next component
```

- Effectiveness calculation:

$$\text{diff\_1} = (\text{mean of total suspicious people–total people stopped})$$
$$\text{effectiveness} = 1/\text{absolute}\ (\text{diff\_1})$$

- Ran over 10000 iteration with 100 travelers each time for both the configurations.

Using the above script for generating travelers with the above choice of parameters, 10.000 iterations with 100 travelers each time were run for each one of the two configurations of **Figures 4** and **5**, and the effectiveness (as defined above) of each configuration was calculated. The results regarding the effectiveness (as defined above) of each configuration are qualitatively summarized schematically in the graph of **Figure 6**.

Figure 6 demonstrates the increase in effectiveness achieved by risk-based security in a BCP using the FhG simulator. The effectiveness of the BCR risk-based configuration 2 clearly surpasses that of the conventional BCT configuration 1. However, the diagram in **Figures 6** and 7 does not include the delays induced by the additional security check stages of the risk-based BSP configuration in **Figure 5**. If we consider these delays, then the operating point of the risk-based BCP not only does it move to higher efficiency but also to higher delays, as **Figure 7** clearly demonstrates.

From **Figure 7** it is clear that there is a competing mechanism between effectiveness (another way of stating "comfort") and delay induced by a risk-based

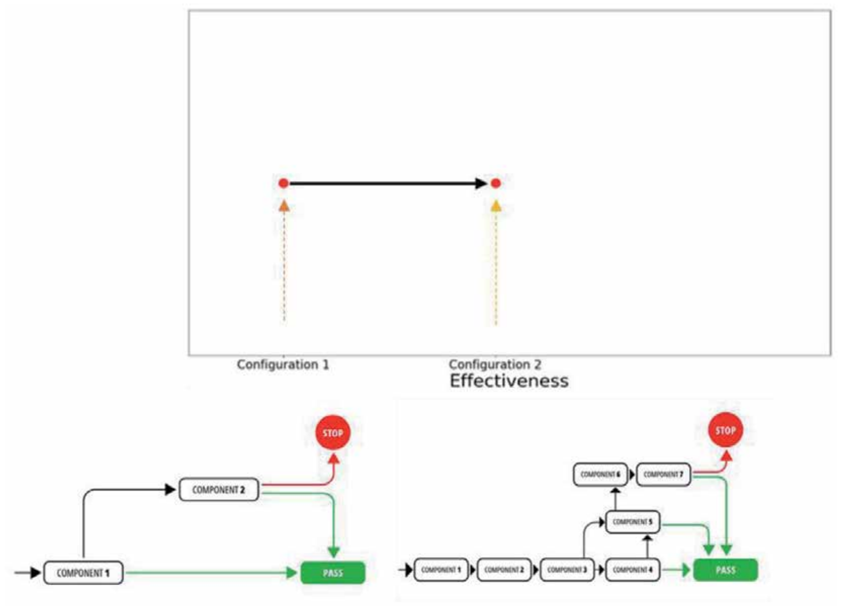


Figure 6.
Effectiveness calculation of a conventional BCP with random security checks determined by importance sampling versus risk-based BCP configuration: Effectiveness increases with the use of risk-based security in a BCP.

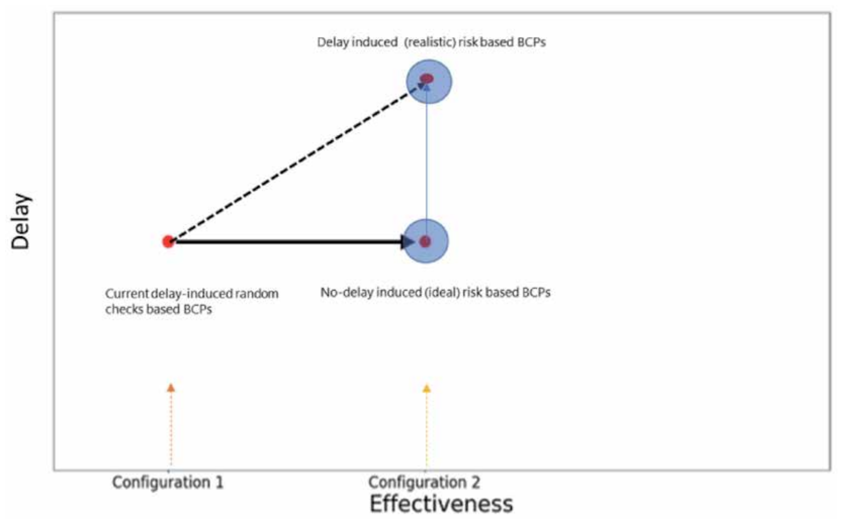

Delay induced (realistic) risk based BCPs
Delay
Current delay-induced random checks based BCPs
No-delay induced (ideal) risk based BCPs
Configuration 1
Configuration 2
Effectiveness

Figure 7.
Effectiveness calculation of a conventional BCP with random security checks determined by importance sampling versus risk-based BCP configuration taking into consideration the additional delays induced by the additional risk assessment stages in configuration 2: Effectiveness increases with the use of risk-based security in a BCP, while induced delays increase as well.

security BCP versus a conventional BCP with randomized tests based on the theory of importance sampling[5]. The aim of the two EU-funded projects FLYSEC and TRESSPASS, coordinated by the author, is for FLYSEC to: (a) demonstrate that there is technology available or can be developed to implement risk-based security in a GDPR compliant way; (b) provide solid evidence of the risk-based security

5 https://www.safeopedia.com/definition/784/safety-sampling

screening as an effective and non-instructive means of providing security with convenience to travelers; and for TRESSPASS to: (c) provide a comprehensive risk-assessment framework for calculating risk systematically in accordance with the TRESSPASS multi threat, multimodal that includes all **four tiers** of the access model, i.e.

1. measures undertaken with third countries or service providers;
2. cooperation with neighboring countries;
3. border control and counter-smuggling measures;
4. control measures within the area of free move,

by taking into account estimates and information about.

Observable aspects of travelers' behaviors, i.e.:

Identity: specific people of which we know that they cause, or will not cause, a threat;

Possession: assets that we know that can be used to generate a threat, e.g. explosives; and.

Hidden aspects of travelers' behaviors, such as:

Capability: people with specific skills with which they can, generate a threat;

Intent: people that have an intent from which a threat can be derived as depicted in **Figures 8** and **9**.

Thus, the aim of the two funded projects, namely FLYSEC and TRESSPASS, is to provide solid evidence and the means for moving the operating point (OP) of a risk-based BCP from the delay induced OP to the no-delay induced OP, or as close to it as possible without inconveniencing travelers and in a GDPR compliant way, as shown in **Figure 10**.

The greatest challenge in risk-based border management is the estimation of the risk for each individual traveler. In TRESSPASS, a framework for modeling risk and a systematic approach of quantifying risk are proposed as follows:

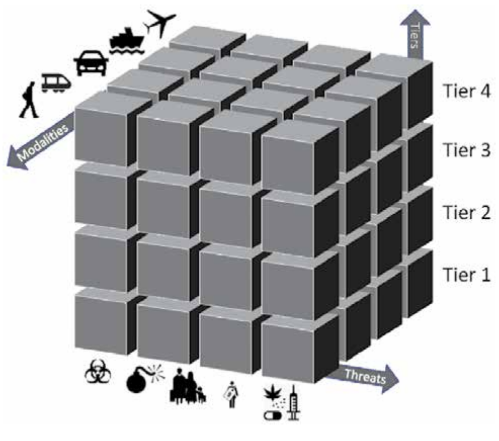


Figure 8.
Multi-modal, multi-tier TRESSPASS risk-assessment model.

- Risk indicators are accurately estimated from available data collected from background information.

- The risk for each traveler is calculated.

- Based on risk, the system adjusts the number and types of security checks required for each traveler, in order to maintain a desired security level while optimizing the security system performance in terms of efficiency, traveler satisfaction and operational cost reduction.

Figure 11 summarizes in a comprehensive visual depiction the risk-based framework used in TRESSPASS [4], and previously introduced in FLYSEC [3]. The framework for risk-based security consists of an extensive use of technologies to

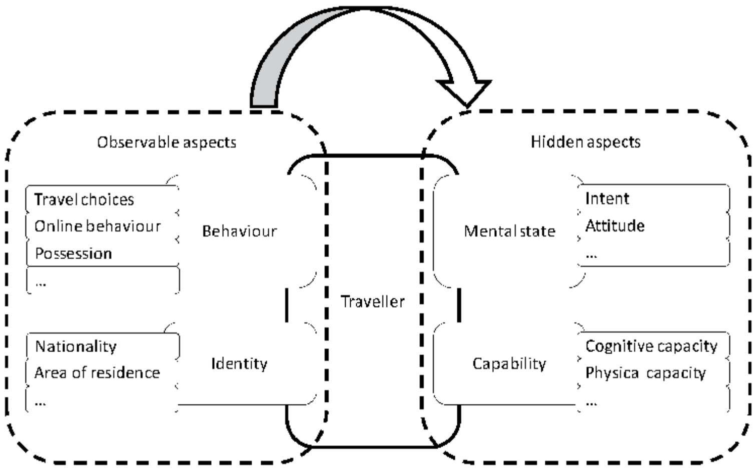


Figure 9.
Observable and hidden risk factors.

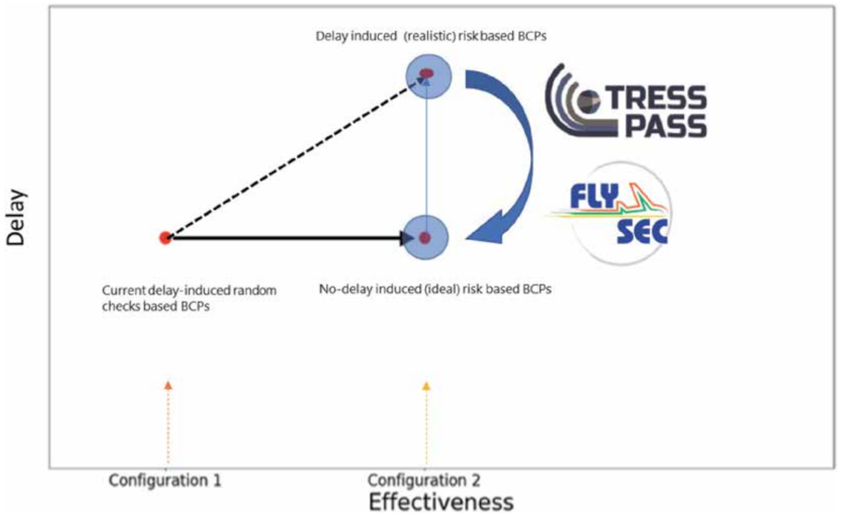

Delay induced (realistic) risk based BCPs
TRESS PASS
FLY SEC
Delay
Current delay-induced random checks based BCPs
No-delay induced (ideal) risk based BCPs
Configuration 1
Configuration 2
Effectiveness

Figure 10.
Moving the operating point of a risk-based BCP to minimizing security check delays is the objective of both FLYSEC and TRESSPASS EU-funded projects.

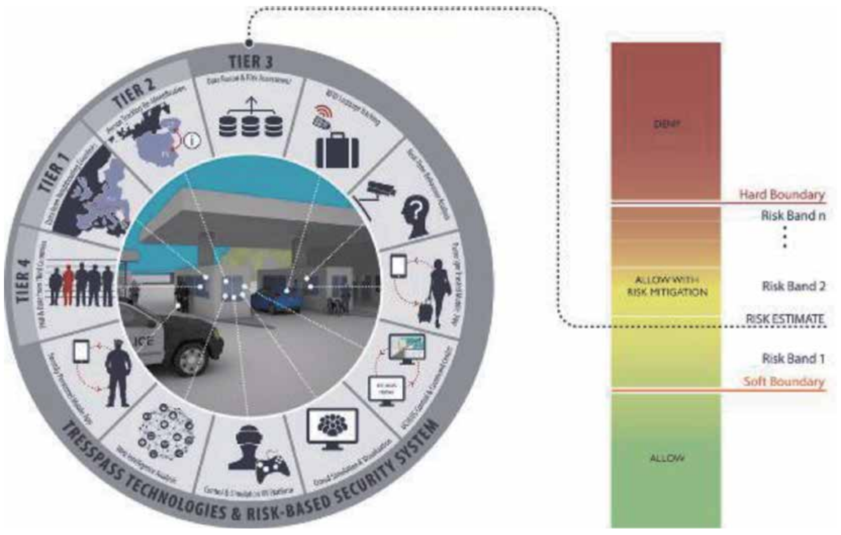


Figure 11.
TRESSPASS comprehensive risk-based security framework.

estimate risk from both Observable and Hidden risk indicators across all four security tiers and heavily tested, both in vivo and in vitro through simulation, in carefully designed pilots across all three BCP modalities: air, land and sea.

Use of simulation in designing, testing, and assessing risk-based security

Paramount to the design and testing alternative designs of risk-based security concepts, technologies and protocols, in order to achieve the increase in effectiveness of BCPs with the parallel reduction of delays, is the use of simulation. iCrowd is an agent-based simulator that can be used to implement and test different risk-based concepts and technologies in a flexible and realistic simulation environment [6]. **Figures 12** and **13** show a photo-realistic virtual reconstruction of an airport used extensively in simulating security scenarios and policies for a variety of projects and pilot use-cases.

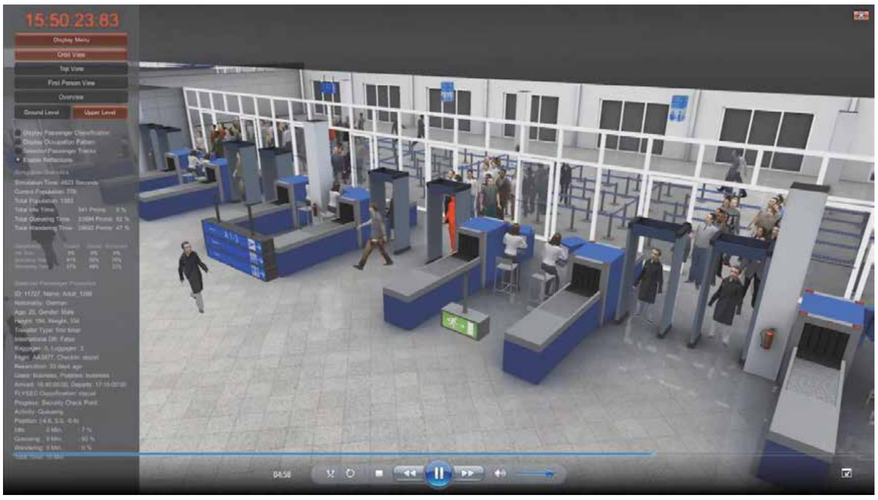


Figure 12.
Photo-realistic, agent-based simulation using iCrowd.

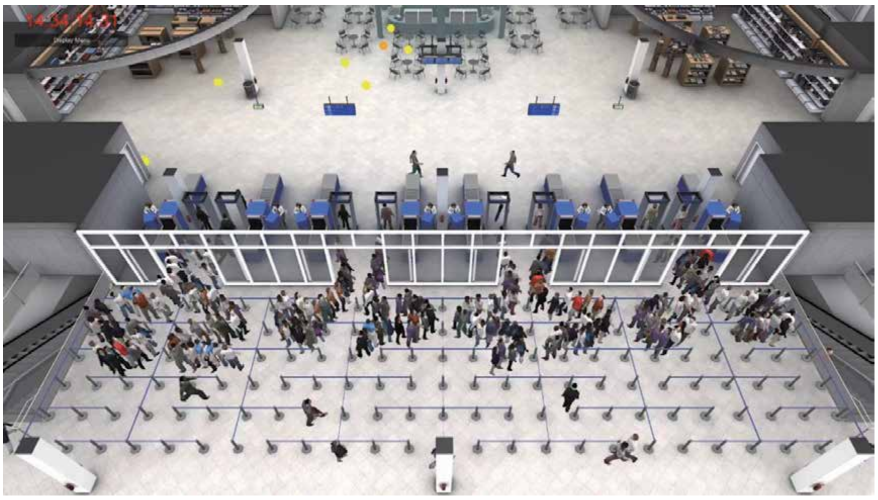


Figure 13.
Queue lanes in a risk-based security checking system: Photo-realistic simulation provided by iCrowd.

2.1 The iCrowd simulator

The iCrowd Simulator is an agent-based simulation platform capable of handling small-scale to large-scale crowds and calculating the change of the status of each participating component depending on dynamic interactions with other entities or the environment during simulation time [7, 8]. It can be utilized in any bounded area, i.e. building interiors and exteriors, stadiums, or any exterior area e.g. public places like squares, open-air festival etc. Currently, it is being used to simulate crowd movement and crowd interactions in general, with the graphical display being optional. As *an agent-based simulation platform, different parameters for each agent can be considered, such as physical, emotional,* vital characteristics regarding the crowd that will be observed (i.e. stress levels, health status), object/obstacle parameters and also environmental parameters that can affect the final solution of each simulated scenario performed.

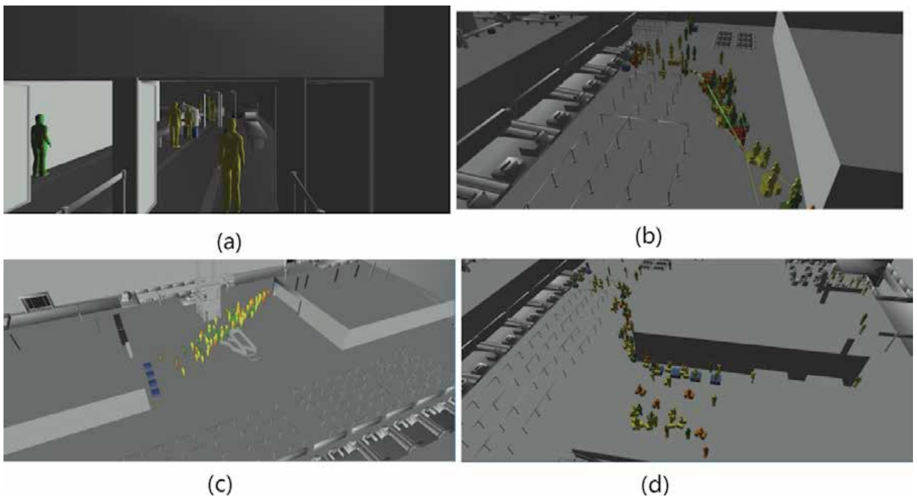


Figure 14.
(a): Aspect of third-person camera. (b): Path planning example: The green line indicates the path the selected agent (displayed are red) is following (c): Travelers enter the airport. The display of hold and hand luggage is turned on. (d): Travelers go through the check-ins. The display of hold and hand luggage is turned on.

iCrowd offers a fully operational flow simulation for travelers and personnel inside an airport, as displayed in **Figures 12** and **13**. It enables the user to define simulation scenarios, it is implementing a sophisticated crowd engine with collision avoidance6 with multiple, different behaviors that can co-exist inside the same simulation. It also supports distributed simulations, operating as an orchestrator. It has been integrated with the C2 Web Portal OCULUS Air to communicate data, such as displaying the position and the movement of simulated entities in real time, and the Fusion and Ingestion Server to update travelers' status accordingly depending on their interactions with the airport's hardware and other security technologies (i.e. Beacons, RFID scanners and RFID tags for carry-on luggage tracking), **Figure 14(a)–(d)**.

3. Means for estimating risk that induce no further delays in the security screening process

This section of the chapter presents and discusses implementable means for assessing risk without inducing additional delays beyond what passengers experience with today's screening process, but instead reduce the time it takes to go through the security screening process by adjusting the level scrutiny in accordance to the perceived risk.

3.1 Anomaly detection from passenger trajectories

If passenger trajectories at an airport, or any BCP by the same token, could be tracked from the moment they enter the airport or the BCP in general, one could conceivably be able to differentiate suspicious looking trajectories from trajectories that would be expected for a passenger and thus classified as normal. Differentiating, however, between normal and abnormal behaviors may be a difficult proposition by itself, let alone that it should be done in accordance with privacy and GDPR regulations.

In the work presented in [9, 10], those two issues were addressed as follows. To develop a privacy and GDPR compliant tracking method, we assumed that passengers are tracked using overhead cameras that identify passengers as point targets from their top-down footprints (silhouettes); the footprints are reduced to a point for each passenger and are tracked across the entire airport area or BCP. In the initial phase of the study in [9, 10], it was assumed that passengers tracking was perfect, i.e. that all passengers' traces as they moved around the airport or BCP area are (anonymously) identifiable and traceable. i.e. that the tracking system has perfect knowledge of the position of each passenger at any time. Although the assumption of perfect knowledge is idealistic, it allows us to get upper bounds on the performance of the tracking system that can be used to make trade off calculations between cost of investment on cameras infrastructure versus the (theoretically) achievable accuracy of the risk calculation.

The difficulty in risk assessment based on trajectories stems from the difficulty in defining what constitutes an abnormal behavior and how it can analytically be described. In the approach in [9, 10] this has been overcome by defining what constitutes a normal (expected) behavior, training the AI (Artificial Intelligence) system to recognize normal behavior and test it with abnormal behaviors to reflect loitering, jittering, and other deviations from expected "normal" behaviors.

Figure 14(a) through **(d)** are snap shots from the native visualizer of the iCrowd simulator simulating an anomaly detection mechanism based on travelers' tracking AI algorithm based on a Recursive Neural Network (RNN) [9, 10]. As discussed above, we assumed that travelers can be tracked anonymously using top-down view cameras in compliance with GDPR and ethics regulations. Based on a model of what constitutes a normal traveler route (trajectory) in an airport (or similarly any other BCP), a convolutional recursive neural network was trained with "normal trajectories" generated by the iCrowd simulator. Once the RNN is trained with "normal trajectories," travelers with "suspicious behaviors" are generated among travelers with "normal behaviors" and the algorithm is tested if it could detect the "suspicious trajectories." In **Figure 14(b)**, the traveler with suspicious behavior is color-coded red. The risk assessment algorithm detects and identifies the suspicious traveler in **Figure 14(d)**.

A complete technical description of the anomaly detection algorithm is given in the references [9, 10]. Next, we summarize the results in [9, 10] in order to demonstrate the possibility of implementing a risk-based security system that monitors traveler risk continually without additional delays that can offset the benefits of the risk-based approach.

3.1.1 Evaluation results

The evaluation of the risk assessment system of [9, 10] is done using the Precision-Recall (PR) diagram, the Receiver Operating Characteristic (ROC) curve, the Confusion Matrix, the F1-score and the Total Accuracy, as defined next:

- Precision = (# of true suspicious behaviors detected)/(# of total labeled suspicious behaviors)
- Recall = (# of true suspicious behaviors detected)/(# of total suspicious behaviors)
- Receiver Operating Characteristic (ROC) curve = Probability of detection versus false alarm probability diagram
- Confusion Matrix = Normal versus abnormal confusion matrix
- F1-score = 2*[(Precision * Recall)/[Precision + Recall] which is the harmonic average of Precision and Recall.
- Total Accuracy = (# of Total Assessments)/(# of Total Cases)

Figures 15–17 illustrate the PR diagram, the ROC curve and the Confusion Matrix respectively. Eqs. 6 and 7 calculate F1-score and Total Accuracy respectively. It should be reminded that the values of all evaluation metrics are defined within the interval [0, 1]. The closer to 1 a value lies, the better the achieved performance. **Table 1** summarizes the values of the recruited evaluation measures. The threshold score derived by the RNN architecture, by maximizing F1-score, is 3.7.

Furthermore, the F1 score (i.e. the harmonic average between Precision and Recall, Eq. (1)), along with the Total Accuracy, Eq. (2), the ROC AUC (Area Under Curve), and Average PR score, are calculated in **Table 1**,

$$F1 - score = 2 \cdot \frac{\text{Pr}ecision \cdot \text{Re}call}{\text{Pr}ecision + \text{Re}call} \tag{1}$$

$$Total\ Accuracy = (\#of\ Successful\ Assessments)/(\#of\ Total\ Cases) \tag{2}$$

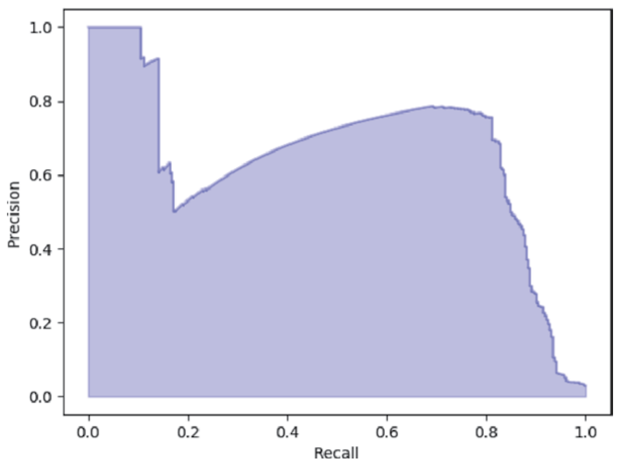


Figure 15.
Precision-recall diagram.

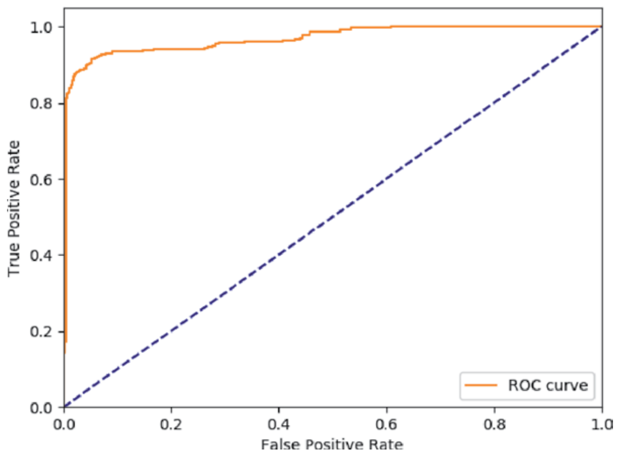


Figure 16.
ROC curve.

3.1.2 Analysis and relaxation of ideal tracking assumptions - experimentation with noisy data

Most of the *false negatives* are abnormal trajectories that cannot easily be discriminated from the normal even by a human operator. Soft thresholding could be used in order to raise alerts for a human supervisor. On the other hand, the main reason for the *false positives* is the fact that airport travelers chose to move in ways that may not necessarily be similar to the normal trajectories. Large airport congestions make the aforementioned phenomenon even more intense.

Although the conditions the risk assessment algorithm was evaluated under assumed perfect knowledge of the traveler trajectories, relaxation of the assumption of perfect knowledge of the traveler trajectories by injecting noise in the position accuracy and/or assuming missing position data, did not have a considerable negative effect on the detection of abnormal trajectories as discussed next.

In order to assess the performance of the anomaly detection algorithm in realistic conditions we introduce noise in the data to emulate the uncertainty in passengers' positions reports. The "noisy data" emulate the inaccuracy in the reports of the

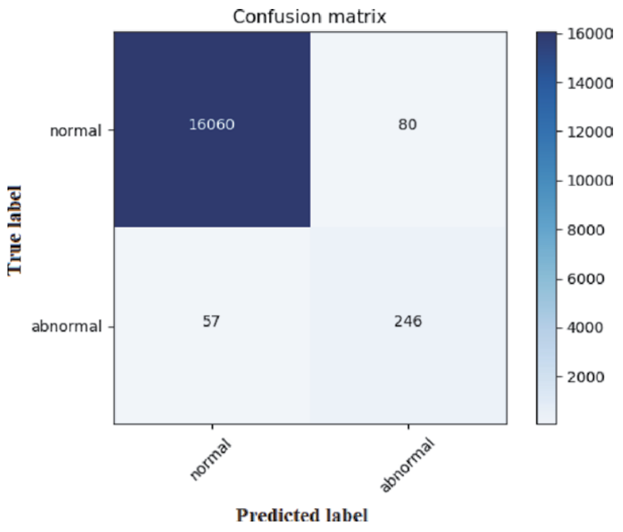


Figure 17.
Confusion matrix.

Measure	Value
Average PR score	0.66
ROC AUC (Area Under Curve)	0.97
F1-Score	0.78
Total Accuracy	0.99

Table 1.
Values of the recruited evaluation measures.

positions of the people in the space. Under realistic conditions, the tracking and risk assessment system will receive data from inaccurate sources, such as cameras, sensors, etc. used to estimate distances, mobile signal strength, etc.

In stark contrast, the iCrowd emulator produces people and their movements, and periodically reports the exact ones (so without noise) their positions in the risk assessment system. During the preprocessing of this data the possibility of the system to add Gaussian noise, the "volume" of which (parameter σ2 of Gaussian noise) is given by the user. This is obviously not intended to never be used in real application, and exists only for experimentation. For examining the behavior of the system under realistic conditions is required noisily data of different intensity.

Noise can enter the system in 2 cases: during training and during testing or actual application. It is known that when training any neural network, it is good to have variety in the data in which the network is exposed so that it is not over-trained. So, it is expected that training with Noisy data can improve the overall performance of the system. During testing or the actual implementation of the system would definitely be better to have perfect data, but unfortunately this is often impossible. In the context of the internship training and validation data were performed with Gaussian noise with σ2 from 0 to 1.9 with step 0.1, testing data with corresponding noise levels, and for each combination they were trained and evaluation of the neural network, and metrics were calculated for each of them. The metrics used were the Receiver Operating Characteristic curve (ROC curve), the Precision-Recall curve (PR curve), and the corresponding Area Under Curve (AUC) scores. These metrics give similar results, in the sense that they are defined in [0,1] with value range [0,1], so the minimum AUC value is 0 and the maximum is 1. Higher value means better true positive and true negative to false positive and false negative ratio. The two metrics generally return similar results, but in our case more weight is given in metric PR, as it offers a better estimate in cases that interest us more the positive class of results, or the results consist of significantly more elements of one of the two classes. Both features apply in the case of this risk assessment system.

During the experiment, networks emerged that failed to find an acceptable solution to why either they were trapped in a local minimum or they encountered the phenomenon of exploding gradient. These cases appeared to be random and independent of the parameters noise, so the network was initialized differently and the training started from the beginning. The experiments were performed using 3 levels of congestion in space, low, moderate, and high, and for each of them 40 neural networks were created, one for each training/testing noise combination mentioned above. For every desired network, 4 independent trainings were conducted and the averages of metrics of interest were kept. The same test data, corresponding to low-to-medium congestion, were used for testing all tested models. The final results are presented below (**Figures 18–20**):

From the ROC AUC score graphs above, it is seen that the models that result in the highest performance correspond to the following noise level in the training data:

For low noise data, the best performing data with AUC = 0.91 corresponds to noise level σ^2 = 0.8 in the training data.

For medium noise data, the best performing data with AUC = 0.96 corresponds to noise level σ^2 = 1.6 in the training data.

For high noise data, the best performing data with AUC = 0.95 corresponds to noise level σ^2 = 1.4 in the training data.

From the above results it is clear that the performance of the networks remains constant when we apply noise to the test data. This implies that, since training completed, the network remains robust and is not affected by data noise, so it can to be used in a real application. Of particular interest are variations that occur when present noise in education data. As mentioned above, training a neuron network usually benefits from the difference in training data, as it helps learn the patterns that appear in the data instead of the data itself. This obviously does not mean that the more noise the better. In every network and for every application there is some optimal noise level that offers the best performance. At cases with low and

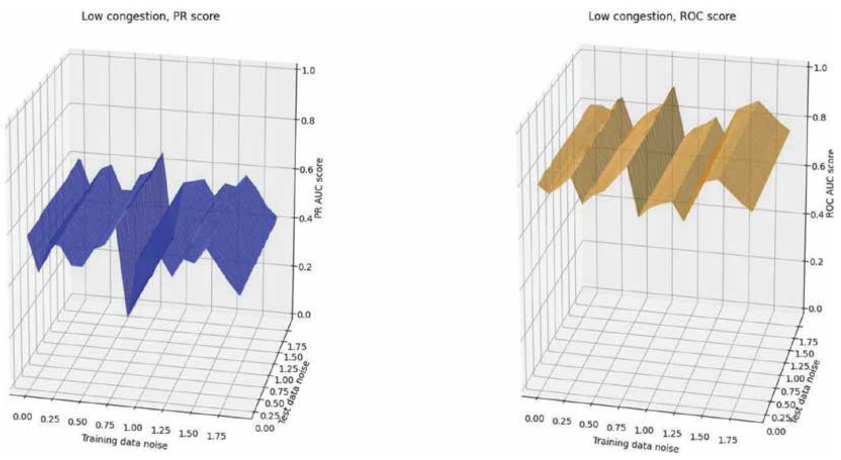


Figure 18.
Training with low congestions data with different noise levels. Testing with low-to-medium congestion data.

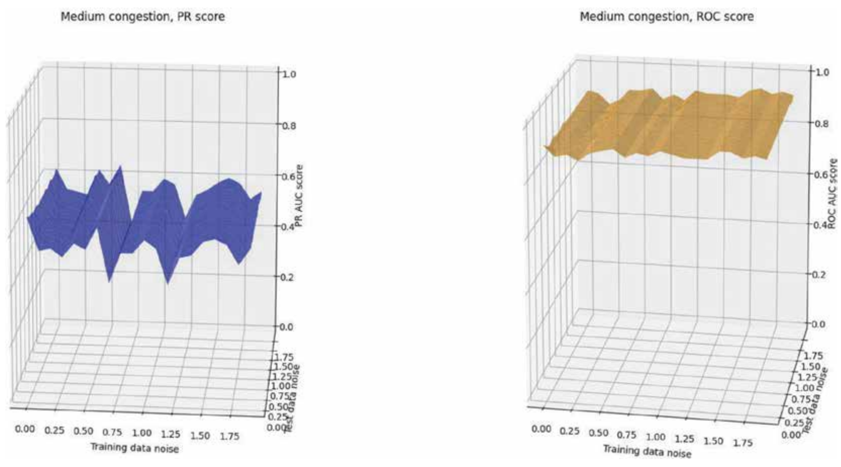


Figure 19.
Training with medium congestions data with different noise levels. Testing with low-to-medium congestion data.

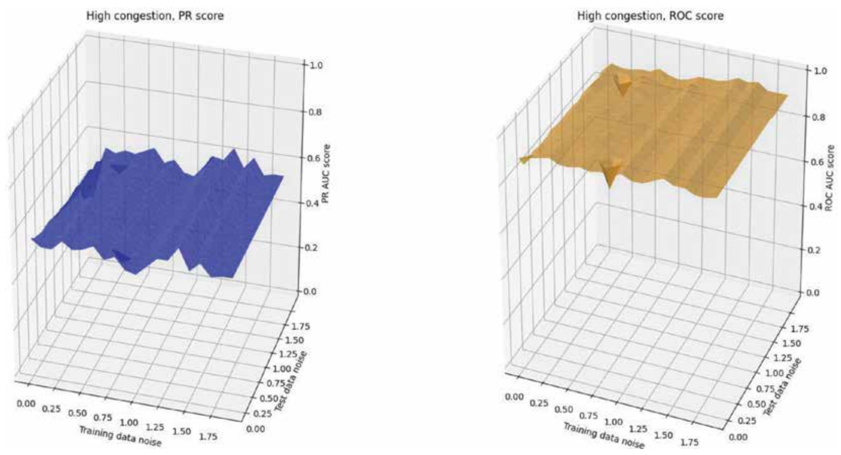


Figure 20.
Training with high congestions data with different noise levels. Testing with low-to-medium congestion data.

moderate congestion it seems that the Gaussian noise with σ2 ~ 0.5–0.8 has the best performance, while for high congestion the training with noise performs better with σ2 ~ 1.4. The variance has not yet been attributed to any of its specific features network, model, training method, or data.

The evaluation results from the performance of the risk assessment algorithm with the iCrowd simulator demonstrates that risk assessment can be done accurately and without necessarily inducing additional delays in the security screening process since the trajectory classification in normal or suspicious is done by overhead cameras while the travelers go about their normal check-in routine at the airport. To that extent, the proposed risk assessment method based on anomaly detection on traveler trajectories can be used to improve the security screening effectiveness while keeping the delay low (or moving the operating point in **Figure 10** from high delay to low.).

Furthermore, the proposed method can be used as a financial investment tool for estimating the cost of acquiring the necessary equipment (in this case overhead cameras) for a certain level (probability of accuracy) before purchasing it, and for performing a trade-off analysis between the cost of acquisition of the necessary equipment and the expected performance improvement in risk assessment. This way, the risk assessment simulator allows to be used as a cost–benefit tool for the analysis of performance of a risk-based security system.

3.1.3 Analysis, relaxation of ideal tracking assumptions and conclusions

In its current form, the work in [9, 10] uses the time series of the coordinates of the trajectories of airport travelers for deep learning. In the future, additional features could be exploited. Such features are the velocity, acceleration and heading of the traveler. Moreover, alternative deep learning architectures could be tested such as the ones that account for contextual anomalies [11]. Furthermore, experiments on real-world data of human trajectories should be conducted. Such data are expected to contain more subtle and sophisticated anomalies. Finally, procedures that degrade data quality and emulate more realistic operational conditions are being implemented in order to test our system in the artificial presence of missing data, noisy data, data association issues, as is the case with data capturing devices operating under realistic operational condtions. Nevertheless, the present work and framework allow security investment decisions on tracking devices and infrastructure to be made by assessing the effectiveness of such an investment through the proposed risk assessment method that envelops the performance of any such system from above by considering ideal tracking conditions through perfect knowledge of all agents' location. The proposed method and framework is currently being extended to cover other border security modalities, such as sea, land as well as multimodal crossing points in the context of the EU-funded TRESSPASS project [4].

In conclusion, a deep learning architecture for real-time risk assessment based on the trajectories of airport travelers as proposed in [9, 10] can be used for assessing risk without interrupting or delaying the flow of passengers at an airport or BCP at large. The architecture implements a deep RNN network and is fully automated. Thus, it is expected to be of great use to the human operators monitoring airport surveillance footages, reducing the potential errors and misjudges. The proposed risk assessment system is tested on a realistic, synthetic data set generated with the iCrowd simulator tailored to data sets representing traveler movements at the Luxembourg airport; however, any airport or BCP could have been modeled and used instead. The experimental results are very promising and they indicate that further security improvements at airport control points are achievable through risk assessment without inducing additional delays. This is due to the fact that the suspicious behavior threshold, derived by the deep learning procedure in [9, 10], lies at such a level so as to capture the malicious behavior while, at the same time, reducing false-positive alerts.

3.2 Risk assessment using a security personnel application and IoT for behavior and event detection through suspicious signs reporting

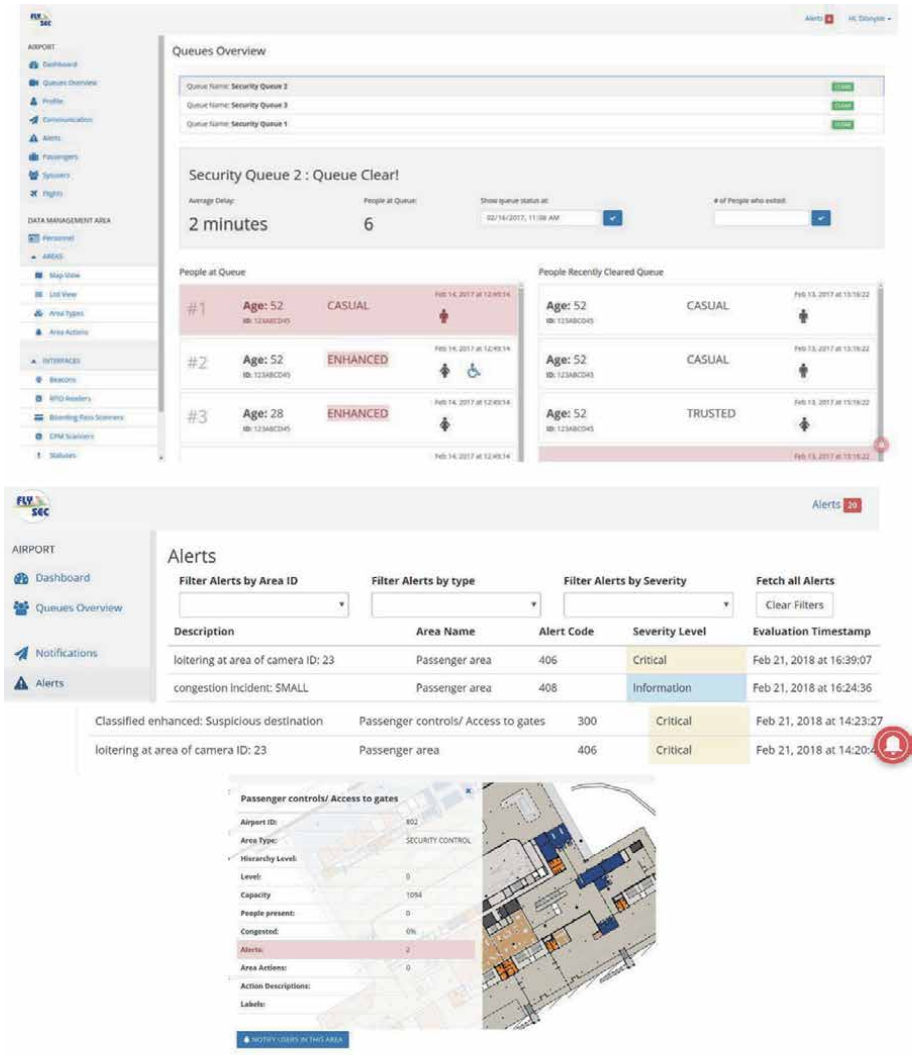
In [3] a GDPR compliant, mobile application was developed to allow security personnel on the floor of an airport, or any BCP, report in real time and with full respect to passengers' anonymity, suspicious behaviors, such as nervousness, unjustifiable sweating, etc., while passengers stand in security check lines. The mobile app works in conjunction with Smart Queue, another enabler of risk-based security [5]. Smart Queue is system that works in conjunction with passengers' ID documents; the system scans the passengers' ID document upon their arrival at the airport, or entry in the BCP, and in any subsequent security queue. This way, Smart Queue not only does it count the number of passengers at a queue waiting to go through security screening, but knows in which position in the queue each passenger stands. This way, the security personnel that uses the security mobile app, needs to identify passengers only by their indexing number in the line they stand when reporting to the risk assessment back office system any suspicious behaviors about them. This way, anonymity of passengers and their personal data protection are maintained by the security mobile app. The information sent this way by the security personnel on the floor is then fused along with all other risk assessment reports about each passenger and the risk estimate is updated. The risk is reported to the security screening system and the passenger is classified in one of the three risk categories, namely green, yellow or red, as mentioned earlier.

In FLYSEC [3], a novel system architecture for Security and Safety surveillance systems that aims to identify adverse events or behaviors which may endanger the safety of people or their well-being has been introduced [12]. Through proper adaptations the system is applicable to a variety of monitoring systems for various critical infrastructures, border crossing points, and other places of interest (e.g., malls, mass transport systems). The proposed architecture depicts an Internet of Things (IoT) platform which comprises a sensing tier, a back – end processing and intelligence tier and a front end for visualization and user feedback tier. In further monitor and surveillance is performed mainly on the back – end intelligence component which consists of two modules: (a) the event detection module combined with a data fusion component responsible for the fusion of the sensors inputs along with relevant high level metadata, which are pre-defined features that are correlated with a suspicious event, (b) an adaptive learning module which takes inputs from security personnel about the correctness of the detected events, and uses it in order to properly parameterize the event detection algorithm. Moreover, a statistical and stochastic analysis component is incorporated which is responsible for specifying the appropriate features to be used by the event detection module. Statistical analysis estimates the correlations between the features employed in the study, while stochastic analysis is used for the estimation of dependencies between the features and the achieved system performance.

3.2.1 System architecture and interfaces

The system architecture is organized basically in three tiers: Sensing components, back–end components, and front – end devices. The sensing components are responsible for acquiring input which is either high or low level heterogeneous data coming from visual sensors (CCD, IR, etc.), biometric sensors (fingerprints, other), audio sensors (microphones), indoor localization equipment (Wi-Fi, beacons, RFID scanners, etc.), document scanners which provide information about visitors (for example travel documents in an airport, or purchase information recorded on personal discount electronic cards), or human reports via terminal devices (e.g. PDAs, mobile phones, tablets, etc.).

Front–end devices are responsible for visualizing information to end–users and assisting their operations (for example official authorities receiving information about detected incidents of great interest, or visitors getting navigation information inside an infrastructure, etc.). Front–end devices consist of official management terminal tools which manage the information collected and processed by the back–end and sensing components and assist personnel operations by providing alerts and notifications about significant events (**Figure 21**), visualizations of infrastructure's layout along with real – time updates about essential points of interest (for example size of queues, sensors viability, crowd distribution, etc.) (**Figure 22**). Moreover, front – end devices include also mobile user devices which operate as a personal assistant to passengers at an airport or a BCP. These mobile devices may provide online and offline services regarding indoor navigation, recommendation



Queues Overview

Queue Name: **Security Queue 2** CLEAR
Queue Name: **Security Queue 3** CLEAR
Queue Name: **Security Queue 1** CLEAR

Security Queue 2 : Queue Clear!

Average Delay: 2 minutes
People at Queue: 6
Show queue status at: 02/16/2017, 11:08 AM
# of People who exited:

People at Queue

#1	**Age:** 52 **ID:** 123ABCD45	CASUAL	Feb 14, 2017 at 12:49:14
#2	**Age:** 52 **ID:** 123ABCD45	ENHANCED	Feb 14, 2017 at 12:49:14
#3	**Age:** 28 **ID:** 123ABCD45	ENHANCED	Feb 14, 2017 at 12:49:14
			Feb 14, 2017 at 12:49:14

People Recently Cleared Queue

Age: 52 **ID:** 123ABCD45	CASUAL	Feb 13, 2017 at 13:16:22
Age: 52 **ID:** 123ABCD45	CASUAL	Feb 13, 2017 at 13:16:22
Age: 52 **ID:** 123ABCD45	TRUSTED	Feb 13, 2017 at 13:16:22
		Feb 13, 2017 at 13:16:22

Alerts

Filter Alerts by Area ID | Filter Alerts by type | Filter Alerts by Severity | Fetch all Alerts | Clear Filters

Description	Area Name	Alert Code	Severity Level	Evaluation Timestamp
loitering at area of camera ID: 23	Passenger area	406	Critical	Feb 21, 2018 at 16:39:07
congestion incident: SMALL	Passenger area	408	Information	Feb 21, 2018 at 16:24:36
Classified enhanced: Suspicious destination	Passenger controls/ Access to gates	300	Critical	Feb 21, 2018 at 14:23:27
loitering at area of camera ID: 23	Passenger area	406	Critical	Feb 21, 2018 at 14:20:4

Passenger controls/ Access to gates

Airport ID:	802
Area Type:	SECURITY CONTROL
Hierarchy Level:	
Level:	0
Capacity	1094
People present:	0
Congested:	0%
Alerts:	2
Area Actions:	0
Action Descriptions:	
Labels:	

Figure 21.
FlySec portal: - intelligent services visualization (upper); – automatic passenger classification (lower).

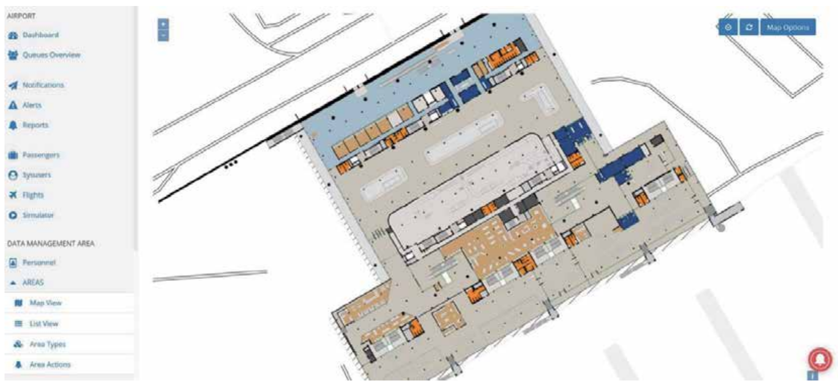


Figure 22.
FlySec portal - layout visualization.

services (for products, point of interests, etc.), notifications and alerts. Finally, via these devices each user may provide a feedback to the system about requests or reports, about incidents that may concern their safety, or public security, or interactions with the system in the context of system automatic personal servicing.

The back–end component contains the intelligence modules which process the input coming from sensing devices and produce high level intelligence and metadata which assist operational personnel, enhance end–users' experience and content management services. These metadata are used either for further processing by fusion algorithms, or presented to end – devices via visualization methods on each end–device. Such metadata concerns directed paths for navigation services, fused high level visual information, or information regarding recommendations, detected incidents or notifications and alerts. Finally, the content management services enable efficient data storing and retrieving operations in a scalable way. The back – end component comprises a Message-oriented middleware in order to interconnect all the sensing and processing component, provides a REST API to front–end devices, supports web platforms interfaces (web – portal) and orchestrates the accurate functionality of the whole system, **Figure 23**.

The core intelligence residing in the "Analytics, Data Fusion and Risk-based Security Server" is presented in Section 3.3. The Data protection, Legal Compliance and Ethics are important aspects that should be taken into consideration in the system architecting process and are analyzed in Section 3.4.

3.2.2 Analytics, data fusion and risk-based security server

Back–end Intelligence component

The proposed system is designed with the aim of enabling automated surveillance of large infrastructures such as airport, shopping malls, other. Such tasks incorporate massive monitoring of infrastructure visitors in real – time. Monitoring operation is based on an Internet of Things (IoT) installation architecture consisting of: (a)

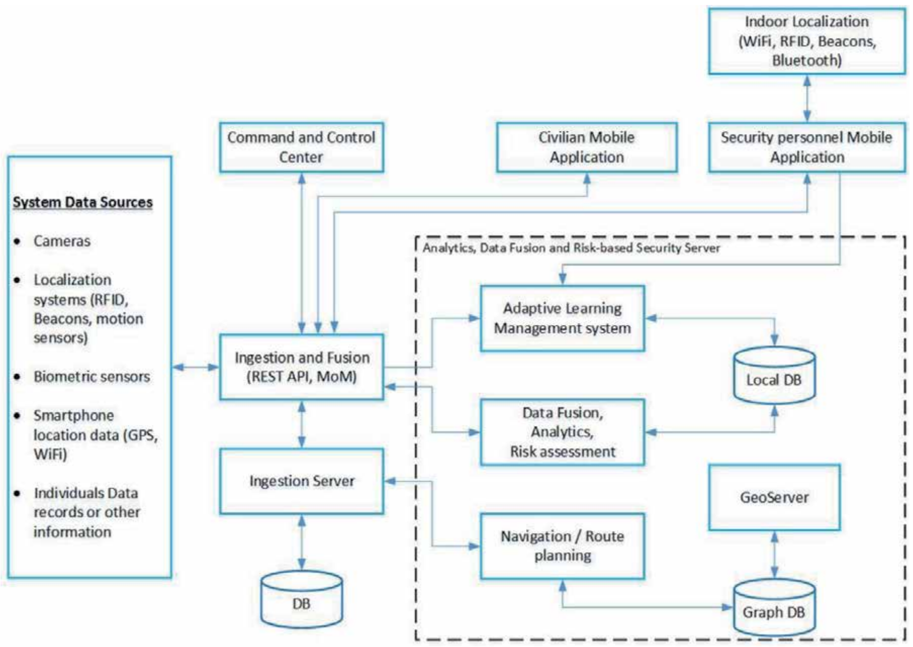

Indoor Localization (WiFi, RFID, Beacons, Bluetooth)
Command and Control Center
Civilian Mobile Application
Security personnel Mobile Application
System Data Sources
Cameras
Localization systems (RFID, Beacons, motion sensors)
Biometric sensors
Smartphone location data (GPS, WiFi)
Individuals Data records or other information
Analytics, Data Fusion and Risk-based Security Server
Adaptive Learning Management system
Local DB
Ingestion and Fusion (REST API, MoM)
Data Fusion, Analytics, Risk assessment
Ingestion Server
GeoServer
Navigation / Route planning
DB
Graph DB

Figure 23.
Reference architecture of the FLYSEC security and safety risk-assessment surveillance system.

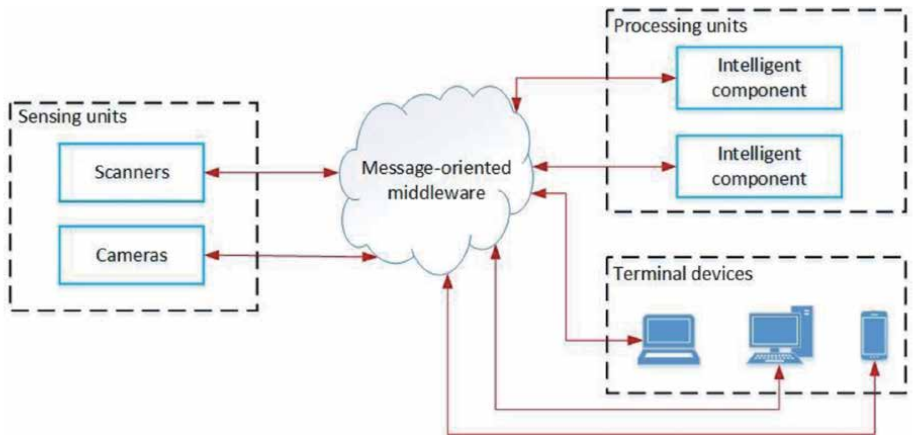


Figure 24.
Back-end intelligence system architecture.

various types of sensing devices such as CCD surveillance cameras, QR/barcode scanners, localization equipment (NFC tags, WiFi beacons, etc), RFID scanners, etc., (b) processing units, both centralized and/or distributed, and (c) terminal devices such as mobile phones/tablets, computers, screens, electric signs, etc. (**Figure 24**).

Each sensor device may pre-process the acquired raw data (distributed processing) and the results are gathered on a central cloud-computing infrastructure consisting of independent but co-operative intelligent component each one dedicated for processing data and producing a specific intelligent response for the system. Moreover, the output is transferred to terminal devices. This processing procedure consists of the following steps (**Figure 25**).

The intelligent services are also responsible for automating the monitoring procedure and enhancing visitors' experience. Therefore, we propose two types of services: (a) *Assistance services* and (b) *Surveillance services*.

Assistance services

These services aim at monitoring visitors' behavior, profile, and interactions and provide information that could facilitate their purpose of visit and indicate services

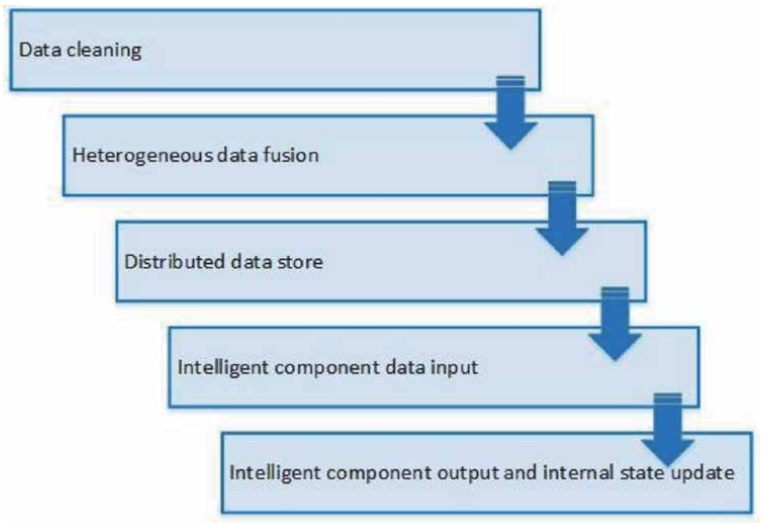

Data cleaning
Heterogeneous data fusion
Distributed data store
Intelligent component data input
Intelligent component output and internal state update

Figure 25.
Flow of data from sensors to the cloud or terminal devices.

that act as added value to visitors and simultaneously promote each infrastructure expectations. Indicatively two representative use cases are Navigation services and Recommendation engine.

- Navigation service corresponds to indoor localization and navigation of infrastructure visitors in order to assist them in reaching their desired points of interest (POI) not only as quickly as possible but also as efficient and desirable as possible by taking into account user requirements (e.g. disabilities, specific demands) and user location. Moreover, the service provide directions to each visitor via their mobile device to various POIs and informs the user in order to assist them reaching their goal of visit (for example provide information about location of various products in a supermarket, or shops in a mall, or provide information about flight departures or gates status in an airport)

- Recommendation engine aims at providing suggestion of POIs or services that take place inside the infrastructure. The engine takes into account the user profile (information that each user provides optionally during account registration), user feedback (comments, rates), user location and contextual information (time, season, POI status) and create recommendations that are estimated to be assistive to user visiting experience but also promoting infrastructure and POIs expectations and benefits.

Surveillance services

These services aim at monitoring visitors' position and behavior and automatically detect incidents of significant interest such as malicious behavior, anomalous crowd trajectory flow etc. This solution is expected to enhance surveillance procedure for large-scale circumstances where it is demanded in real time, the accurate surveillance of a massive crowd. Indicatively we suggest two surveillance services: Suspicious unattended luggage incidents detection and suspicious visitor loitering detection.

- Unattended luggage incidents detection aims at monitoring in parallel both visitors and the luggage they carry. Such monitoring could be approached either using CCD cameras and approximately detect abandoned luggage for a long period of time, or by tagging luggage (for example using RFID tags) where using RFID scanners in co-operation with visual sensors (CCD cameras) and human reports (official surveillance personnel), estimate potential unattended luggage incidents. Moreover, in order to monitor visitors' position, we propose the use of indoor localization techniques using mobile devices in order to have an approximation of visitors' location that willingly allow it, and in addition visual sensors and human reports as well, in order to increase system's awareness of crowd location. Fusion of such information shall be exploited by machine learning algorithms, which result to a coarse grain estimation of visitors' luggage abandonment.

- Suspicious loitering detection aims at monitoring visitors' location and in real–time detect anomalous visitors' trajectories or positions that could be suspicious for malicious purposes. Such components may incorporate visual sensors (CCD cameras), human reports and mobile devices localization techniques (Wi-Fi beacons, NFC tags).

Data fusion and Risk-based assessment

The Data Fusion unit inside the Analytics, Data Fusion and Risk-based Security Server aims to perform Hard and Soft fusion of heterogeneous data [13–15] available from disparate sources of information such as physical sensors ("hard" data) and human resources ("soft" data). Hard data fusion refers to the combination of raw information from multiple sources so as to achieve more accurate estimations of the desired parameters (position, speed, other). To this end, a variety of theoretical tools, such as Signal processing techniques, Kalman filters, Sequential Monte Carlo methods, etc., can be used. On the other hand, soft data fusion usually applies on textual information (e.g., from humans' reports, social networks, Internet, other) which has to be further processed using methods such as Information retrieval, Natural Language processing, and Semantic knowledge representation. Moreover, in this unit, Decision level fusion techniques could be applied using Evidence theory [14–16], Fuzzy Logic [17], 2-tuple Linguistic representation models [18, 19], and reinforcement learning methods [20, 21].

The Risk-based assessment unit is responsible for the classification of events and individuals into security classes according to their risk severity level. The unit exploits behavior and event indicators and their corresponding weights estimated in the ALMS system, intelligence generated in the Back–end Intelligence component, and any useful information from the system's data sources in order to generate alerts and notifications to the Command-and-Control (C2) center if the risk severity level exceeds predefined thresholds.

Adaptive learning management system

A security and safety monitoring system has to detect, evaluate, and classify, in an efficient and timely manner, behaviors and events of interest. To achieve this critical need, the algorithmic parameters used in the "Analytics, Data Fusion and Risk-based Security Server" have to be initialized and adaptively adjusted to handle changes in the monitoring environment. To this end, the use of an Adaptive Learning Management System (ALMS) which will exploit new and accumulated information is essential. An ALMS system can be applied for instance to iteratively adjust the Risk Assessment classification thresholds and the weights of the behavioral and event indicators or to recognize correlations between indicators, events, and behaviors in order to optimize the classification process and improve the efficiency of the system. An example of such an optimization approach could be the selection of a reduced number of indicators for event identification.

For the development of automated procedures able to estimate correlations, optimize selected parameters under certain criteria, and extract reduced dimensional feature vectors for Behavior and event detection the ALMS system demands efficient methodologies and algorithms. These methodologies and techniques can cover a wide area of theoretical tools including Machine Learning, Factor Component Analysis, Statistical methods, Time series analysis, Optimization theory, Sparse clustering, Fuzzy Logic, and other [18–21].

As shown in **Figures 23** and **26**, the ALMS unit receives input from i) the system's database which includes data from system's data sources, outputs of Data Fusion, Analytics, and Risk assessment unit, and optimization criteria and constraints and ii) the Security personnel Mobile App which is then used for the training of the applied algorithms. The ALMS stores its output in the system database, making it accessible to other units, and creating a continuous feedback loop of information gathering, learning, and adapting to security threats as they evolve.

The Factor Component Analysis component performs Factor Analysis on features/indicators denoting individual characteristics which affect the categorization of individuals in security-threat levels. Factor analysis is used to reduce the dimensionality of a correlation matrix that contains features/indicators describing a specific event or behavior. Factor Analysis does that by producing new general variables, called "factors", incorporating inside them, the initial features/indicators according to a condition of high inter-correlation between the newly

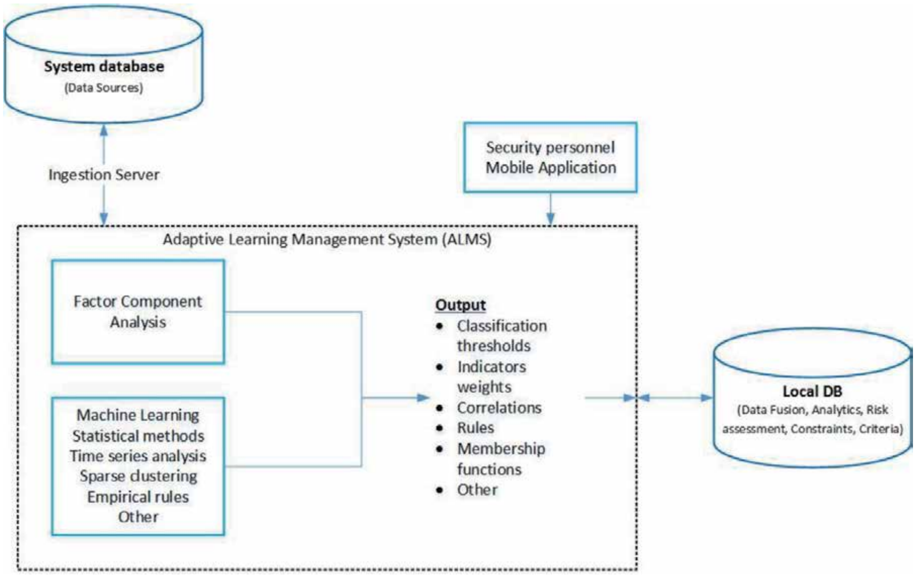


Figure 26.
The adaptive learning management system architecture.

emerged general variables ("factors") and the initially presented specific variables [19, 22].

The system needs to analyze the input feed and result to its outcome, taking into account however, environmental factors regarding system's efficiency, explicit policies that should by adopted or exceptions that should be applied, that are related to specific locations (e.g., restricted areas) or specific human profiles. Such information usually is returned to the system in the form of a generic asynchronous qualitative feedback (for example insisting user discards of system's outcomes, or exception to system's rules) that should be assimilated in real – time.

The system should be able to receive environmental feedback and adapt its operation to the current circumstances and requirements. Therefore, we propose a two-mode adaptation, an offline and an online. The offline adaptation regards a system initialization, responsible for translating human – understandable requirements to algorithms' parameterization. The online adaptation should track environmental feedback for each action of set of actions (policies) produced by the system and adapt algorithms' behavior in order to fulfill system's requirements.

In this case we propose the implementation of reinforcement learning techniques where environmental feedback should be encoded to quantitative measures of rewards, **Figure 27**.

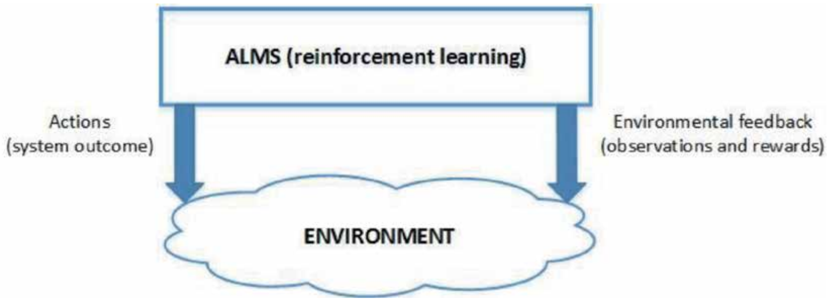


Figure 27.
Online ALMS.

3.2.3 Data protection, legal compliance and ethics

Security and safety management systems and their data fusion and intelligent analytics capabilities require substantial data collection and processing in order to offer the best possible awareness and decision support to C&C operators, field personnel and first responders. Especially in the context of homeland security, privacy and data protection is often seen through the typical trade-off model perspective, requesting the public to give up –in the best case knowingly- on particular rights over the control of their personal data. However, such systems should not be based and developed on exceptions or operate only in extraordinary circumstances, the latter being very inefficient. With the latest guidelines of EU General Data Protection Regulation (GDPR), principles of data minimization and privacy by design will shift from best practices into a much more regulated form.

The proposed system is in line with these principles, following a "by design approach" in terms of data protection and ethics. Data collected are structurally separated from identifiable information, and identification occurs only upon the logged and explicit intervention of a human operator when truly needed. By assessing risks on real time, the system itself has the advantage of performing data minimization through early elimination of lower risk cases. On the front end and field, privacy enhancing technologies and smart sensors are also preferred and selected. E.g. smart visual sensors with on-board processing capabilities can filter out data before sending it over the wire and to the server for processing. Moreover, the system has been designed to include specific safeguards to protect individuals against discrimination, stigmatization and unduly prohibition of access to goods and services. Defined in [23], the system adopts these definitions and extends them to all protected grounds as defined in the Charter and the Treaty of Amsterdam, taking also into account the proposal for the horizontal directive that extends the context of EU non-discrimination law and prohibits discrimination "on grounds of sex, racial or ethnic origin, age, disability, sexual orientation, religion or belief". In this context, Fairness and bias detection algorithms are applied to the adaptive learning management system while the human operator remains in control of the final enforcement following any automated decision making process. Intelligent behavior analytics can further support the case where security risks are based and calculated on how a person acts on the scene and not any discriminatory background information.

A subject of past and current research, assessing the societal acceptance of surveillance and security solutions comes with its own challenges. Acceptance is based on multiple parameters, individual perceptions and sometimes misconceptions and individual practices which may not be in line with the expressed concerns [24]. The proposed system and the overall risk-based security paradigm, is based on the positive fact that the vast majority of people have no malicious intent. The system focuses on the unknown and high-risk cases, intending to shift the current practices from annoying horizontal and disruptive processes to seamless and unobtrusive security. The combination of privacy and ethics by design along with the ethical and unobtrusive treatment set the parameters for a system with high acceptance, positive public perception and trust.

4. Conclusions

In this chapter we discussed the concept of risk-based security, the possible trade-off between increased convenience for passengers from risk-based security and the delays induced by additional checks needed for establishing each passenger's risk. We also presented a number of technologies, systems and applications that can be used for assessing risk at an airport or BCP without inducing additional delay as the discussed approaches estimate risk on-the-fly while passengers either walk around the airport or BCP from entrance to security check points or BCPs, or queue up in a security line awaiting to go through security checks. All methods discussed are GDPR and ethics compliant, thus they can be implemented in accordance to privacy and ethics regulations. Furthermore, the novel system architecture for Security and Safety monitoring systems introduced in [3] has been presented. The proposed system aims to identify adverse events or behaviors which may endanger the safety of people or their well-being having the ability to adapt in the surveillance environment changes. The dynamic adjustment of the algorithmic parameters adopted in various units of the system such as intelligence, and Risk assessment, makes it possible to monitor security threats as they evolve. Thus, the proposed scheme provides the potential of a high-performance system both in terms of the detection interval as well as in terms of the performance accuracy offering the capability of a timely and efficient response to abnormal events and behaviors.

Acknowledgements

The research described in this paper has been supported by the following research contracts:

"**FLYSEC**: Optimizing time-to-FLY and enhancing airport SECurity," Programme: Horizon 2020, European Union Grant Agreement No. 653879, Duration: 01/05/2015 - 31/07/2018, http://www.fly-sec.eu.

"**TRESSPASS**: Robust Risk Based Screening and Alert System for Travelers and luggage," Grant Agreement No. 787120, Call: H2020-SEC-2016-2017-2, https://www.tresspass.eu/The-project.

The author would also like to acknowledge the use of some material from the Refs. [5–10, 12, 14]. He co-authored in collaboration with his colleagues whose names appear in these references.

Author details

Stelios C.A. Thomopoulos
Integrated Systems Laboratory, Institute of Informatics and Telecommunications, National Center for Scientific Research "Demokritos", Greece

*Address all correspondence to: scat@iit.demokritos.gr

References

[1] European Commission: "Smart Borders Package": https://ec.europa.eu/home-affairs/what-we-do/policies/borders-and-visas/smart-borders_en

[2] IATA Checkpoint of the Future: http://www.iata.org/pressroom/pr/Pages/2011-06-07-01.aspx

[3] "FLYSEC: Optimizing time-to-FLY and enhancing airport SECurity," Programme: Horizon 2020, Contract No. 653879, 01/05/2015–31/07/2018, funding organization: European Union, http://www.fly-sec.eu.

[4] "TRESSPASS: Robust Risk Based Screening and Alert System for Travelers and luggage," Programme: Horizon 2020, Contract No. 787120, Call: H2020-SEC-2016–2017-2, funding organization: European Union, https://www.tresspass.eu/The-project.

[5] Stelios C. A. Thomopoulos, Dimitris M Kyriazanos, Andreas Zalonis, 'FLYSEC: A comprehensive control, command and Information (C2I) system for risk-based security', Signal Processing, Sensor/Information Fusion, and Target Recognition XXVII, v. 10646, International Society for Optics and Photonics, 25/6/2018.

[6] V. Kountouriotis, S. C. A. Thomopoulos and Y. Papelis, "An agent-based crowd behaviour model for real time crowd behaviour simulation." Pattern Recognition Letters/Pattern Recognition and Crowd Analysis, vol. 44, pp. 30–38, (2014).

[7] Kountouriotis V. I., Paterakis M., and Thomopoulos S. C. A., "iCrowd: agent-based behavior modeling and crowd simulator," SPIE DSS 2016 - Defense, Security and Sensing, Convention Center Baltimore, Baltimore, Maryland, United States, 17–21 April, (2016).

[8] Daveas S., and Thomopoulos S. C. A., "Embedding a Distributed Simulator in a Fully-Operational Control & Command Airport Security System," in Proceedings Volume 10646: Signal Processing, Sensor/Information Fusion, and Target Recognition XXVII, June (2018).

[9] Stelios C. A. Thomopoulos, Stelios Daveas and Antonios Danelakis, "Automated real-time risk assessment for airport passengers using a deep learning architecture," Proceedings Volume 11018, Signal Processing, Sensor/Information Fusion, and Target Recognition XXVIII; 110180O (2019) https://doi.org/10.1117/12.2519857 Event: SPIE Defense + Commercial Sensing, 2019, Baltimore, Maryland, United States, 2019.

[10] Giorgos Bouritsas, Stelios Daveas, Antonios Danelakis, Stelios CA Thomopoulos, "Automated Real-time Anomaly Detection in Human Trajectories using Sequence to Sequence Networks," 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), p. 1–8, 2019/9/18.

[11] Rikard Laxhammar, Göran Falkman, and Egils Sviestins, "Anomaly detection in sea traffic - a comparison of the Gaussian Mixture Model and the Kernel Density Estimator," in [IEEE International Conference on Information Fusion], (2009).

[12] Konstantinos Georgios Thanos, Constantinos Rizogiannis, John M. A. Bothos, Dimitris M. Kyriazanos, Andreas Zalonis, Stelios C. A. Thomopoulos, "A novel architecture for behavior/event detection in security and safety management systems", Proc. SPIE 10200, Signal Processing, Sensor/Information Fusion, and Target Recognition XXVI, 102000R (16–19 Aril

2018). Proceedings Volume 10646, Signal Processing, Sensor/Information Fusion, and Target Recognition XXVII; 106460V (2018) https://doi.org/10.1117/12.2307079. Event: SPIE Defense + Security, 2018, Orlando, Florida, United States

[13] Klein, L. A., [Sensor and Data Fusion: A Tool for Information Assessment and Decision Making], SPIE press (2004).

[14] Thomopoulos, S. C. A., "Sensor Integration and Data Fusion," Proc. SPIE 1198, Sensor Fusion II: Human and Machine Strategies, 178–191 (1990).

[15] Raol, J. R., [Multi-Sensor Data Fusion with MATLAB], CRC Press (2009).

[16] Shafer, G., [A Mathematical Theory of Evidence], Princeton University Press, Princeton, NJ (1976).

[17] Pedrycz, W. and Gomide, F., [An Introduction to Fuzzy Sets: Analysis and Design], MIT Press (1998).

[18] Martínez, L. and Herrera, F., "An overview on the 2-tuple linguistic model for computing with words in decision making: Extensions, applications and challenges," Information Sciences 207 (1), 1–18 (2012).

[19] Rietveld, T. and van Hout, R., [Statistical Techniques for the Study of Language and Language Behaviour], (1993).

[20] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D. and Kavukcuoglu, K., "Asynchronous Methods for Deep Reinforcement Learning," Proc. of the 33rd International Conference on Machine Learning, New York, NY, USA (2016).

[21] Sutton, R. and Barto, A., [Reinforcement Learning: An Introduction], MIT Press, (1998).

[22] Field, A. P., [Discovering Statistics Using SPSS for Windows], (2000).

[23] D. Potoglou et al, "Literature Review of Approaches for Measuring Preferences with Respect to Privacy, Security and Surveillance," The Privacy & Security – Research Paper Series, (2014).

[24] FRA Handbook of EU non-discrimination law, available online (Accessed March 2018): http://fra.europa.eu/sites/default/files/fra_uploads/1510-FRA-CASE-LAW-HANDBOOK_EN.pdf.

Chapter 8

Application of Deep Learning Methods for Detection and Tracking of Players

Marina Ivasic-Kos, Kristina Host and Miran Pobar

Abstract

This chapter deals with the application of deep learning methods in sports scenes for the purpose of detecting and tracking the athletes and recognizing their activities. The scenes recorded during handball games and training activities will be used as an example. Handball is a team sport played with the ball with well-defined goals and rules, with a given number of players who can participate in the game as well as their roles. Athletes move quickly throughout the field during the game, change position and roles from defensive to offensive, use different techniques and actions, and very often are partially or completely occluded by another athlete. If artificial lighting and cluttered background are additionally taken into account, it is clear that these are very challenging tasks for object detectors and trackers. The chapter will present the results of various experiments that include player and ball detection using state-of-the-art deep convolutional neural networks such as YOLO v3 or Mask R-CNN, player tracking using Deep Sort, key player determination using activity measures, and action recognition using LSTM. In the conclusion, open issues and challenges in applying deep learning methods in such a dynamic sports environment will be discussed.

Keywords: handball, object detector, object tracking, action recognition, person detection, deep convolutional neural networks, YOLO, mask R-CNN, LSTM, DeepSort, Hungarian algorithm, optical flow, STIPs

1. Introduction

Computer vision (CV) is a compelling field of Artificial intelligence that develops the theory and methods by which information about the real world can be automatically extracted and analyzed from image data. Image data can be in many forms, such as image, video, depth image, multi-camera views, or multidimensional data from a medical scanner.

The objective of CV is to model the real world or to recognize objects from digital images enabling computers or devices to "see", interpret, manipulate, analyze, and understand what was seen and draw conclusions about the properties of the 3D world based on a given image or a sequence of images [1].

Basic CV tasks are image classification, segmentation, similarity calculation and object localization. Recognition of objects present on the scene and their features (e.g., shapes, textures, colors, sizes, spatial arrangement,) is often prerequisite for more complex CV tasks such as image retrieval, image description, object detection, object tracking, action recognition, image or scene analysis, and image understanding [2].

In all tasks, the starting point are the image features that carry important information and need to be extracted and processed in order to generate new information and conclusions. The image features can be divided into low-level features such as corners, edges, or contours that can be extracted with relatively simple image operations, and high-level features that require domain-knowledge to get structured information related to the object or action being taken [3].

Feature extraction can be described as a pre-processing step to remove redundant parts from the data and keeping the key information for accomplishing the task. Some well-known features that can be extracted are Optical flow for extracting motion information, Histogram of Oriented Gradients (HOG) and Silhouette for extracting shape information, Space–Time Interested Points for extracting interest points, etc.

To accomplish typical CV tasks, Image processing and Machine learning (ML) play an important role. Image processing is focused on low-level features and manipulation of image data for normalizing photometric properties of visual data, removing digital noise, data augmentation, etc., and is not concerned with understanding the content of visual data. However, when it comes to interpret the content and draw conclusions about the image to automate CV tasks, the most important fields are ML and its subfield Deep learning (DL) [2].

Before DL, computer vison tasks required a lot of coding and manual effort to define the features that can be extracted from images, with little automation involved [4]. With DL methods such as Convolutional Neural Network (CNN) [5], much of that work related the features to be extracted can be inferred automatically from data. Even though many features can be extracted automatically in the CNN framework for different tasks, manual feature extraction can still be useful for either augmenting the automatically extracted features or perform other tasks such as temporal segmentation of video or detection of active players.

Typical CV tasks such as object detection, object tracking, and action recognition, the tasks we will focus on the most here, are supervised learning tasks (**Figure 1**). Supervised learning relies on labeled ground truth data, based on which the learning algorithm infers the mapping between the raw data and the desired labels in the training stage. Thus, a prerequisite for supervised learning is data preparation, which includes data collection and labeling, pre-processing and feature extraction, followed by splitting data into a training and testing set, then selecting an appropriate learning algorithm and model structure for the specific task. After training and validating the model, the model needs to be tested and the obtained results need to be compared with the ground-truth data to evaluate the

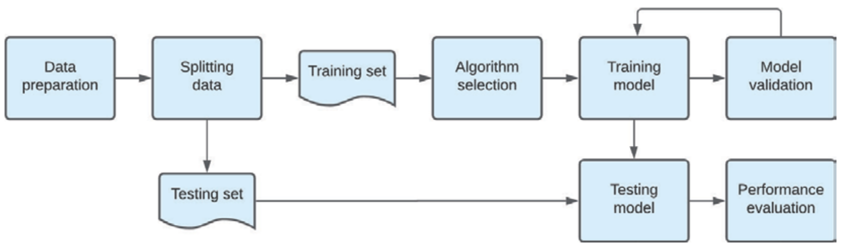


Figure 1.
Supervised learning process.

performance. The performance evaluation is represented with different metrics appropriate for the specific task.

CV tasks can be implemented for image and video analysis in different domains, including the sports domain. Various CV techniques can be very useful for all parties interested in analyzing the game, including the coach, the reporters, the referee team, the physiotherapists and others, for making decisions about an occurred event, for monitoring and comparing the performance of each individual player, for choosing a strategy, for fast automatic analysis of video materials captured during a match or practice and the like.

In this chapter, the focus will be on handball, a team indoor sport played with a ball by two teams with seven players, one of whom is the goalkeeper. To analyze handball sports videos, different CV tasks can be combined. For example, the object or person detection can be applied to detect the players on the field, the object tracking to follow the players' movements across the field, and action recognition to analysis of the players' performances.

In the next sections, a created dataset for handball will be presented, and then the simple CNN architecture and typical measures for evaluation of model performance are described. In the following sections, CV tasks will be described and implemented in the context of handball. Object detection with YOLO and Mask R-CNN is presented in Section 5, object tracking with Hungarian algorithm and Deep SORT in Section 6, and action recognition using LSTM model in Section 7. Applications of optical flow and spatiotemporal interest points for temporal segmentation and active player determination are presented in Section 8.

2. The dataset

The handball dataset used for the following experiments was recorded during a handball school, where participants were young handball players and their coaches.

The dataset consists of high-quality video recordings of practices and matches, filmed in a sport hall or in an outdoor handball field, without additional scene preparation or player instruction to preserve real-world conditions. The recordings were made using different stationary cameras positioned on the left or right border of the field on a tripod at 1.5 m, or from the spectator's viewpoint at the height of approximately 3.5 m and the distance from the filed limit of 10 m. The recordings are in full HD resolution (1920x1080) and contain from 30 to 60 frames per second.

The dataset is quite challenging with a cluttered background, a variable number of players at different distance from the camera, who move fast and often change direction, are often occluded with another player, have jerseys of similar color to the background, etc.

The data needs to be prepared, processed and labeled for each specific task that will be considered here, so that domain- and task-specific models can be made by either training from scratch if there is sufficient data, or preferably, by tuning an existing model for a similar task using examples from the new domain.

For the player and ball detection task, 394 training and 27 validating images were extracted from the videos in the handball dataset and manually labeled, to form the PBD-Handball dataset [6].

To obtain the ground-truth data for the player tracking task, a subset of videos from the handball dataset was first processed using the YOLOv3 object detector, then with the DeepSORT tracker to bootstrap the annotation process, and lastly manually corrected. The total duration of the annotated dataset, named PT-Handball, prepared for this task is 6 min and 18 s [7].

For the action recognition task, parts of the videos containing the chosen actions were extracted from the whole handball dataset to get a subset of 2,991 short videos that were then labeled with one of the 10 action classes, or the Background class where action is not happening. This dataset is referred to as PAR_Handball.

3. Convolutional neural networks

Typical models used today for image classification and object detections tasks are based on Convolutional Neural Networks (CNNs), since they are adapted to solve the problems of high-dimensional inputs and inputs that have many features. The CNN network consists of a number of convolution layers, after which the network has been named, the activation and pooling layers, and one or more fully connected layers at the end of the network [8], (**Figure 2**).

The convolution layer refers to a mathematical operator defined over two functions with real value arguments that give a modified version of one of the two original functions. The layer takes a map of the features (or, in the first layer, the input image) and convolves it with a set of learned parameters resulting in a new two-dimensional feature map. The sets of learned parameters (weights and thresholds) are called filters or kernels. Each filter is a 2D square matrix, small in size compared to the image to which it is applied (equal depths as well as the input). The filter consists of real values that represent the weights that need to be learned, so that the output feature map contains useful information such as a particular shape, color, edge in order to give the network good results.

The pooling layer is usually inserted between two successive convolution layers to reduce a map resolution and increase spatial invariance or network insensitivity to minor shifts such as rotations, and translations of features in the image. The pooling layer also reduces memory requirements for network implementation. The most commonly used pooling methods are arithmetic mean and maximum, but several other pooling methods are also used in CNN architecture, such as Mixed Pooling, Stochastic Pooling, Spatial Pyramid Pooling and others [8].

The activation function defines the output of a node given an input or set of inputs. In its simplest form, this function is binary and represents the action potential of neurons by propagating the output value of the neuron or by stopping it. There is a broad range of univariate functions of linear combination of the input variables acting as CNN activation functions such as linear activation functions, jump functions, and sigmoidal functions. The jump and sigmoidal functions are a better choice for neural networks that perform classification tasks while linear functions are often used in output layers where unlimited output is required. Newer architectures use activation functions, typically Rectified Linear Unit (ReLU), behind each layer.

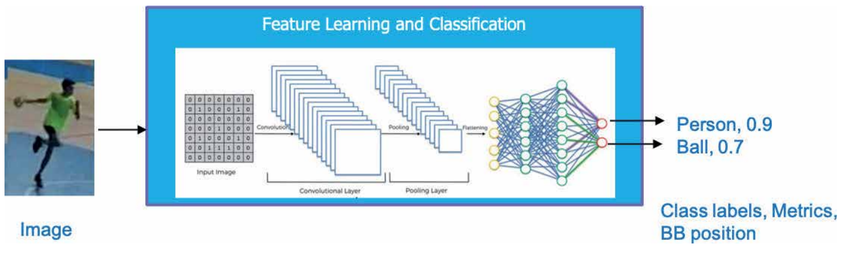


Figure 2.
Simplified CNN architecture.

A fully connected layer is the last layer in the network. The name comes because of its configuration: all neurons are linked to all the outputs of the neurons in the previous layer. Fully connected layers can be viewed as special types of convolution layers where all feature maps and all filters are 1 x 1.

Network hyperparameters are all parameters needed by the network that have to be set before the network is provided with data for learning. The hyper-parameters in convolutional neural networks are the learning rate, the number of epochs, the number and kind of network layers, the activation function, the initialization weights, input pre-processing and the error function.

Selecting the structure of the CNN network for feature extraction plays a vital role in object detection because the number of parameters and types of layers directly affect the memory requirements, speed, and performance of the detector.

In this paper, two types of CNN-based networks, YOLO and Mask-RCNN, have been used for object detection, while for the task of action recognition, the CNN network is not used on its own, but it forms a part of the more advanced LSTM network as the feature extractor.

4. Evaluation measures

The performance of object detectors is usually evaluated in terms of accuracy, recall, precision, and F1 score [9], for a given confidence threshold. The same measures can also be used for evaluation of the action classification task.

The detections are deemed true positive when the intersection over union of (IoU) the detected bounding box and the ground truth box is greater than 0.5. The IoU measure is defined as the ratio of the intersection of the detected bounding box and the ground truth (GT) bounding box and their union, see **Figure 3**.

Since the confidence threshold controls the tradeoff between recall and precision, Average Precision (AP) measure is frequently used to evaluate the performance of the detectors. The AP is the area below the precision-recall curve which is calculated for every class by varying the confidence threshold. To get the mean Average Precision (mAP) value, mean of AP values of all classes is calculated.

Since there is no single measure that can uniquely describe the complex behavior of trackers, several measures are used to evaluate the tracking performance. These measures are the number of identity switches (ID), identification precision (IDP), identification recall (IDR) and the identification F1 (IDF1) measures [10].

An identity switch occurs when an object that was assigned an ID j in previous frames, gets a new id k, $k \neq j$ in a subsequent frame. The IDF1 measure focuses on how long a target is correctly identified, regardless of the number of mismatches. It is the ratio of correctly identified detections over the average number of ground-truth and computed detections.

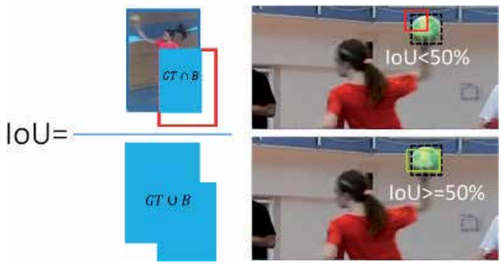


Figure 3.
Visualization of intersection over union (IoU) criteria equal to or greater than 50%.

5. Object detection

The task of object detection is to find instances of real-world objects in images or videos. A detected object is typically marked with a bounding box and labeled with a corresponding class label and classification confidence value. Thus, object detection includes both the problems of finding the location of the object on the scene and of classification for predicting the class to which the object belongs to.

In case of player detection, the object detector should be able to overcome challenging conditions such as variable number of players, different player positions, varying distance of the player from the camera, the possibility of changing shape and appearance of players in time, presence of the blur of due to the speed of the movement, occlusion, shadows of artificial and external light, as well as cluttered background.

Nowadays, the focus in object detection is on CNNs that have been extended to be able to both detect and localize individual objects on the scene. In the following subsections, two different object detectors YOLO and Mask R-CNN are described with a corresponding experiment of player and ball detection.

5.1 YOLO

YOLO is a detection algorithm based on a single-stage CNN architecture that can detect multiple objects in an image in real-time. The main idea is to predict bounding boxes and confidence values for grid cells into which an image or frame is divided. In the cases when an object is spread across more than one grid cell, the holder of its prediction will be the center cell.

There have been four versions of YOLO since it was first published. In the original version, the network architecture has 24 convolutional layers with two additional fully connected layers. The purpose of the convolutional layers is the feature extraction, while for fully connected layers to calculate the bounding boxes predictions and probabilities. The bounding box predictions and class probabilities are associated with grid cells so that if an object occupies more than one cell, the center cell will be designated to be the holder of prediction for a particular object [11].

In the next version, YOLOv2 [12], five convolution layers were replaced with max-pooling layers, and instead of the fully connected layers, predefined anchor boxes are introduced. In the training phase, to define the anchor boxes, YOLOv2 uses k-means clustering on ground-truth bounding boxes where boxes translations are relative to a grid cell.

YOLOv3 [13] is the third version of the YOLO object detector. It consists of 53 convolutional layers of (3×3) and (1×1) filters with shortcut connections between layers (residual blocks) used for feature extraction. The last convolutional layer predicts the bounding boxes, the confidence scores, and the prediction class. It predicts possible bounding boxes at three different scales using a structure that is similar to feature pyramid networks. In this way three sets of boxes are predicted at each feature map cell for each scale, to improve the detection of objects of different sizes.

5.1.1 Player detection with YOLO

Player and ball detection performance of the YOLOv3 detector was tested on the handball dataset using two different models. The reference model, further marked as Y, is the pre-trained YOLOv3 model with 608 x 608 input image size with weights pre-trained on the COCO dataset and no additional training. The pre-trained model contains the person and sports ball among other classes from the COCO dataset. Transfer learning [14] was used to avoid training the models from the beginning.

The second model (YBP) was trained using transfer learning, on PBD-Handball part of the dataset. The input image resolution was increased to 1024 x 1024 from 608 x 608 of the original model and the model was trained for approximately 80 epochs. **Figure 4** shows an example of detection results for the "person" class.

To evaluate the performance of a model, the average precision (AP) metric of both classes and the mean average precision are used and shown on **Table 1**.

The best results for ball detection in terms of AP were achieved with the YPB model, which was trained on additional examples for both ball and person class and had an increased input image size. A small amount of training data can significantly improve detection results as can be seen in the example of ball detection which improved for 23%. The achieved results are satisfactory given the demanding environment but are not sufficient for commercial application, so the training dataset should be increased.

5.2 Mask R-CNN

Mask R-CNN [15] is a two-stage CNN that can not only detect and localize multiple objects simultaneously present in the image, but also provides a segmentation mask of the objects, that is, assigns a membership value to each of the pixels belonging to the object. The first stage of the network is a region proposal network that finds the regions of the image that are likely to contain objects (regions of interest, RoI) and proposes candidate object bounding boxes. A sliding window is applied to the feature map to examine the probability whether there is an object class or a background in the examined region. Then, bounding boxes and masks are generated with the corresponding confidence values for all possible classes.

In the second stage, there are two parallel branches of the network, a fully convolutional branch for predicting the segmentation masks and a fully connected branch used on each RoI for classification and for adjusting the proposed box size.

There are similar networks like R-CNN, Fast R-CNN, Faster R-CNN [16] on which Mask R-CNN is based to look up for object detection purpose.

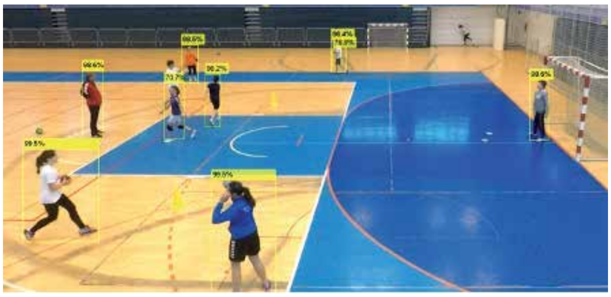


Figure 4.
Player detections in handball scene with YOLOv3 (bounding boxes with confidences).

	PBD-Handball		
Model	**Ball AP**	**Person AP**	**mAP**
Y	13.53	66.13	39.83
YPB	35.44	63.77	49.61

Table 1.
Evaluation of the object detector.

Object Detector/Measure	Inference time / frame	Recall	Precision	F1
YOLOv3	0.04 s	68%	95%	79%
Mask R-CNN	0.3 s	76%	98%	85%

Table 2.
Results of player detection with mask R-CNN and YOLOv3.

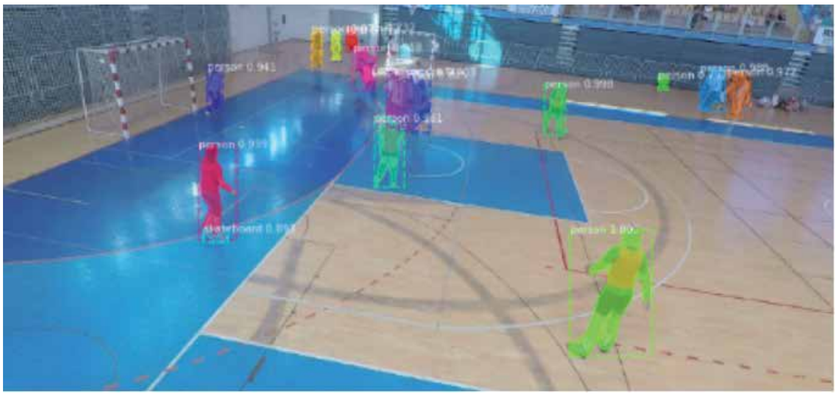


Figure 5.
Player detections obtained with mask R-CNN in a handball scene (bounding boxes with segmentation mask and confidence value).

5.2.1 Player detection with mask R-CNN

The performance of the Mask R-CNN for player detection was tested on the PBD-Handball dataset using the standard Resnet-101-FPN network configuration with pre-trained parameters on the COCO dataset. For player detection experiment, only the bounding boxes that refer to the "person" class were considered.

To obtain a good balance of high detection rates and low false positive detections, detections with confidence values below a threshold experimentally set to 0.55 were discarded. The detector performance was evaluated in terms of recall, precision, F1 scores and inference time per frame (using the NVIDIA 1080ti GPU). The results and comparison with the YOLOv3 detector are shown in **Table 2**. Detection was considered as true positive when the intersection of the detected bounding box and the ground truth box was above 50%.

One handball scene with the bounding boxes, class confidence value, and segmentation masks obtained with Mask R-CNN is shown on **Figure 5**.

It can be concluded that the results of both the YoloV3 and Mask R-CNN detector are good enough to be used for further analysis of player performance. However, the YOLOv3 detector is much faster, so it can be used not only for offline analysis of recordings, but also for real-time detection, at the cost of somewhat reduced recall. The detection results could be improved if more data is used, but the performance depends on the number and size of the players on the scene, the contrast between a player and a background, illumination, etc.

6. Object tracking

Tracking of handball players in video is an example of a Multi-Object Tracking (MOT) problem, where the goal is to track both the position and the identity of multiple objects present in video, so that the same unique ID is assigned to each object in every frame it appears. In an ideal case, in every video frame, all the present players should be detected in their correct position and a unique ID for each player, that stays the same throughout the video, should be assigned. This is a difficult task as many players can be on the field, from 14 to 25, depending if it is a practice or a match, and every one of them needs to be tracked. Furthermore, players can leave and re-enter the camera field of view, move very quickly, often change directions, occlude each other and wear similar clothes, or clothes with similar color as the background [17].

Thanks to improvements in performance of object detectors, and thanks to the ability to deal with challenges such as cluttered scenes or dynamics of tracked objects, tracking-by-detection has become a leading paradigm for MOT.

When using tracking-by-detection, the tracking algorithm relies on the object detector to detect and locate the objects on the scene in each frame, while the role of the tracking algorithm itself is reduced to the problem of associating the detections across frames that belong to the same object. To do so, the tracker may use the information about bounding boxes obtained by object detection, such as their dimensions, the locations of their centroids, the relative position to the boxes in previous frames, or some visual features extracted from the image.

6.1 Hungarian assignment algorithm

The Hungarian algorithm [18] solves the problem of finding the globally optimal assignment of IDs to detected player bounding boxes, with respect to some cost function that is defined for an individual assignment. Here the cost function is defined only in terms of the parameters of the bounding boxes detected by the object detector in the current and previous frame, without using any visual features extracted from the video frames. Its value depends on the Euclidean distance of each detected object's bounding box centroid from the predicted centroid of an object in the track, and on the size difference of the bounding box and the last assigned bounding box to the same track.

Formally, the assignment cost $\mathrm{d}(b,k)$ of assigning a bounding box b with the centroid C_b and area P_b to the k-th track with the predicted centroid C_{k+1}' is:

$$\mathrm{d}(b,k) = w \cdot d_2\left(C_b, C_{k+1}'\right) + (1-w)\left|P_b - P_{k-1}\right| \tag{1}$$

$$w \in [0,1],\ k \in N$$

where P_{k-1} is the area of the last bounding box assigned to the track with the ID k.

For the prediction of the centroid location C_{k+1}', last known position of object centroid in the track k is used, so $C_{k+1}' = C_{k-1}$.

Moreover, a unique track ID is assigned to each detected bounding box whose detector confidence is higher than a set threshold during the initial assignment of bounding boxes to tracks. Afterwards, whenever the number of detected objects exceeds the number of currently active tracks, new tracks are created and initialized using the unassigned object's bounding box.

An existing track is considered inactive when no detections are assigned to that track for several frames. Once a track is marked inactive, no further detections are added to it, so if an object later reappears or is detected again, it will get a new track ID and will be considered a new object.

6.2 Deep SORT

Deep SORT [19] is a tracking algorithm that builds upon the Hungarian algorithm, adding the appearance information about the tracked objects into consideration when associating new detections with previously tracked objects. The appearance information is particularly useful for re-identifying players that were occluded or have temporarily left the scene. As in the previous case, a unique track ID is assigned to each bounding box within the first frame, and the Hungarian algorithm is used to assign the new detections to existing tracks so that the assignment cost function reaches the global minimum.

The cost function consists of the spatial distance $d^{(1)}$ in form of Mahalanobis distance of the detected bounding box from its position predicted according to its last known position, and a visual distance $d^{(2)}$ that compares the appearance of the detected object with the history of appearances of the tracked object. Formally, the cost function $c_{i,j}$ of assigning a detected object j to a track i is given by:

$$c_{i,j} = \lambda d^{(1)}(i,j) + (1-\lambda) d^{(2)}(i,j) \tag{2}$$

where λ is a tunable parameter that determines the relative influence of the spatial distance $d^{(1)}$ and the visual distance $d^{(2)}$.

The spatial distance $d^{(1)}$ is given by the expression:

$$d^{(1)}(i,j) = \left(d_j - y_i\right)^T S_i^{-1} \left(d_j - y_i\right) \tag{3}$$

where y_i and S_i represent the mean and the covariance matrix of bounding box observations for the ith track, and d_j is the jth detected bounding box.

The visual distance $d^{(2)}$ is is given by the expression:

$$d^{(2)}(i,j) = \min\left\{1 - r_j^T r_k^{(i)} | r_k^{(i)} \in \mathcal{R}_i\right\}, \tag{4}$$

where r_j is the appearance descriptor extracted from within the jth detected bounding box, and $\mathcal{R}_i$ is the set of the last 100 appearance descriptors r_k^i associated with the ith track.

The $d^{(2)}$ measure uses the cosine distance between the jth detection and a number of detections already assigned to ith track, so if a visually similar detection was previously seen, the distance will be low.

The appearance descriptors are extracted using a wide residual neural network comprising two convolutional layers followed by six residual blocks that output a 128-element vector, and then normalized to fit within a unit hypersphere so that the cosine distance can be used. The network was pre-trained on a person re-identification dataset of more than a million images of 1261 pedestrians. The appearance information helps with re-identification of objects that have not been tracked for some time because of missed detections, because they were under occlusion or because they have briefly left the scene.

New tracks are formed whenever there are more detections in a frame than there are existing tracks or when a detection cannot be assigned to any track, because its spatial or visual distance is too far from any existing track. The maximum allowed $d^{(1)}$ and $d^{(2)}$ distance when an assignment is still possible is set with tunable thresholds.

6.3 The player tracking experiment

The tracking of players previously detected with the YOLOv3 detector using pre-trained tiny-yolo weights and confidence threshold set to 0.5 was performed on the PT-Handball dataset with the Hungarian algorithm and Deep SORT [20].

An example of a tracking situation in sequential frames when occlusions occurred is shown in **Figure 6**. The numbers above the bounding boxes represent the tracking ID of each player. The shown situation is quite demanding, resulting in rather unstable and inconsistent tracking in the selected frame. The Hungarian algorithm successfully tracked one of four players, and two of the players got new IDs after occlusion, while one player has switched ID with another, so that 807 got previously existing ID 812 (**Figure 6**, top row). Deep SORT managed to track correctly all four players (**Figure 6**, bottom row).

Since the best results were obtained with the Deep SORT algorithm, its ability to assign the correct IDs to detections is analyzed in more detail using the common MOT evaluation measures. The results are shown in **Table 3**.

For each player that should be tracked, the identity switches caused, 5 to 6 additional tracks on average, so there are 5 times more tracks than in ground-truth annotated data. Furthermore, a large number, precisely 1483, of identity switches are present, due to a relatively large number of players in the video that move fast, exit the camera field view, frequently change positions, and occlude each other.

The number of players that are simultaneously present in the frame obviously affects the tracking performance, and according to the IDF1 measure, the players can be correctly identified for 24,7% of the time.

Tracking mistakes can be attributed to several factors. As in all tracking-by-detection algorithms, the accuracy of tracking is greatly influenced by the accuracy of the object detector. If a player is inaccurately detected, the tracking will be inaccurate as well. Furthermore, the scale of an object, occlusion, and the similar color of the players' clothes with the background often cause tracking problems. To overcome these problems, in further work, multiple camera systems will be investigated [21], which can also allow for a more robust generation of a top-view trajectory [22].

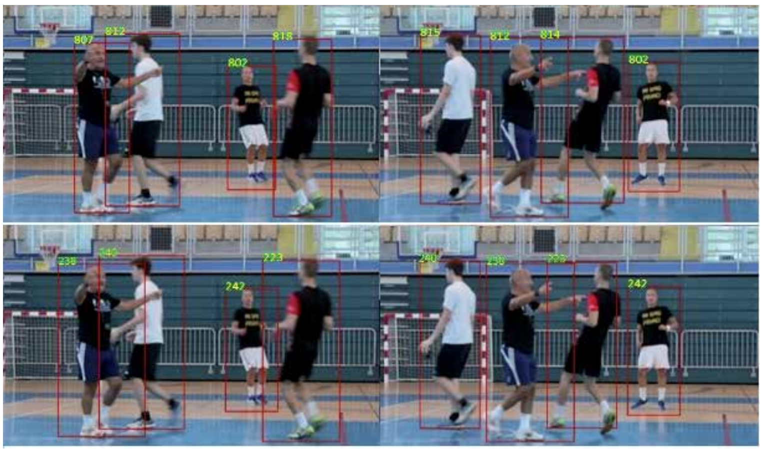


Figure 6.
An example of tracking situation with occlusion. Top: Hungarian algorithm, bottom: Deep SORT. The left and right frames are 1 second apart.

Measure	Value
#tracks in the ground truth	279
#tracks	1554
Identity switches	1483
IDF1	24.7%

Table 3.
Performance evaluation of deep SORT.

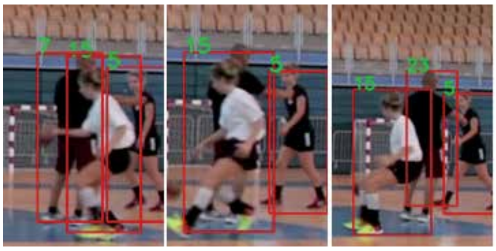


Figure 7.
Problem of re-identification after occlusion.

In **Figure 7**, the top row shows the problem of re-identification after occlusion, and the bottom row the problem of identity switch due to small scale and similar colors.

7. Action recognition

The goal of action recognition is to infer which action takes place in a set of image or video observations. Some simple actions, like eating or cutting, could be recognized using just a single frame, but actions are mostly much more complex and take place over a period of time, so they need to be analyzed across consecutive video frames.

Here, for recognition of handball actions, a simple long short-term memory (LSTM)-based artificial recurrent neural network is used.

During the handball game, every player is moving around the field performing different actions with a ball, such as shot, jump-shot or dribbling, or without the ball, such as running or defending. Some actions are performed by more players such as passing the ball or crossing.

7.1 LSTM

Unlike CNNs and other so-called feedforward neural (FFNN) networks, recurrent neural networks (RNNs) [23] have connections that feed the activations of an input in a previous time step back into the network, to influence the output for the current input. These activations from the previous time step are held potentially indefinitely in the internal state of the network, so the temporal context is not limited to a fixed window that could be used as an input to a FFNN. This property makes RNNs especially appropriate for modeling sequences, such as text or a sequence of video frames in action recognition.

Long Short-Term Memory (LSTM) [24] is a type of recurrent neural network (RNN) designed to model temporal sequences and their long-range dependencies more accurately than conventional RNNs. In the recurrent hidden layers, the LSTM contains special units called memory blocks. Those units contain memory cells, that have self-connections storing the temporal state of the network, and gates, which are special multiplicative units that control the flow of information. The flow of input activations into the memory cell is being controlled by the input gate, while the output gate controls the output flow of cell activations into the rest of the network. The forget gate scales the internal state of the cell before adding it as input to the cell through the self-recurrent connection of the cell, thus causing the adaptive forgetting or resetting the cell's memory.

7.2 The action recognition experiment

In the experiment handball actions from the PAR-Handball dataset are considered: Throw, Catch, Shot, Jump-shot, Running, Dribbling, Defense, Passing, Double-pass, and Crossing. An example of jump-shot action is shown in **Figure 8**. The action consists of a sequence of different phases of jump-shot action from running, take-off, flight, throw, and landing that are captured on different video frames.

Different actions can take different amounts of time to perform, so the average number of frames in a video depends on the action class it belongs to, as shown in **Figure 9**.

Only Throw and Catch actions are significantly shorter (two or three times) than the other action classes that have a duration of around 60 frames on average.

Because these two actions are also parts of the more complex ones, like Passing, the model is trained once with all 11 classes and once with 9 classes, excluding Throw and Catch.

The model selected for action recognition is a LSTM-based network with one LSTM layer with 1,024 units, followed by a dropout layer with 0.5 dropout, one

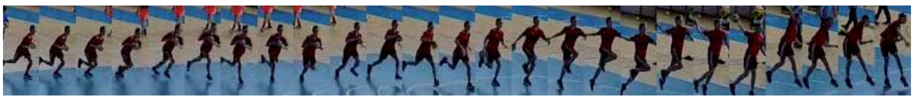


Figure 8.
Active player collage for jump-shoot action.

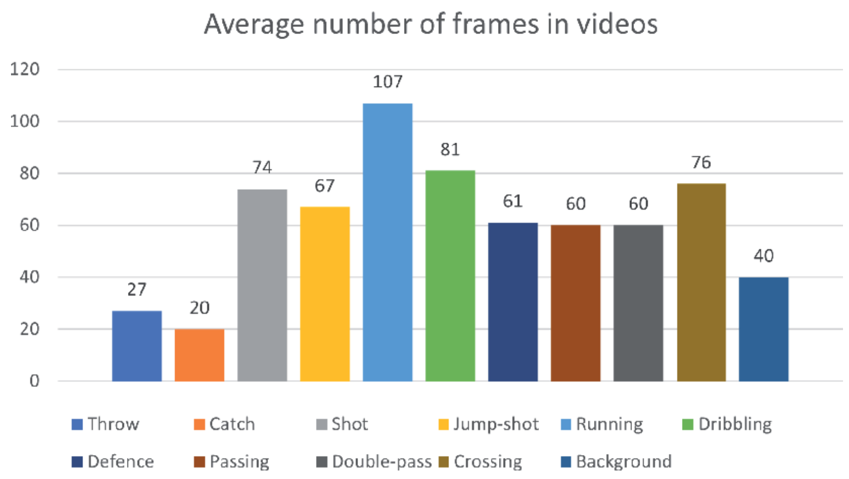


Figure 9.
Average number of frames per action from the handball dataset.

fully connected layer with 512 neurons, also followed by a dropout layer with 0.5 dropout rate. and the output layer with 11 neurons.

The input to the LSTM consists of a sequence of features extracted from video frames using the InceptionV3 [25] network with the ImageNet [26] re-trained weights as the starting point. The model is trained with the Adam optimize with a learning rate of 0.00001 and decay of 10–6 for up to 100 epochs, stopping early if the validation loss does not improve for more than 20 epochs.

Different frame selection strategies and different input size of sequences from 20 to 80 were used to train the model, because actions might have most distinctive characteristics in different parts of the sequence.

In videos containing more frames than expected, the chosen number of frames were selected consecutively from either beginning, middle or the end of the video, or from the whole video by decimation, i.e., by skipping some frames at regular intervals. Conversely, copies of existing frames were inserted between frames to extend the number of frames.

The action recognition results obtained by the described LSTM model, considering the frame selection strategies and different class numbers are shown in **Figure 10** in terms of validation accuracy.

Having to classify a smaller number of classes can generally be considered a simpler task, so, as expected, the models trained on 9 classes have better results on average than the models trained on 11 classes. However, the best result overall of 70.94% is obtained for the model with 11 classes and 45 frames taken from the middle of the sequences. This is possibly due to the overlap of some actions that make them more difficult to recognize, as the actions Throw, and Catch that are parts of other actions such as Passing, Double-pass, Crossing, Shot and Jump-shot. Closely behind with 70.55% is the model trained on 9 classes with 20 frames in the middle, and in the third place is the model trained on 9 classes with last 45 frames with 70.47% validation accuracy.

Taking into consideration only the number of input frames and ignoring the number of classes or frame selection, the best results are obtained with 45 frames followed by 20 frames. In most cases the additional frames in the sequence do not

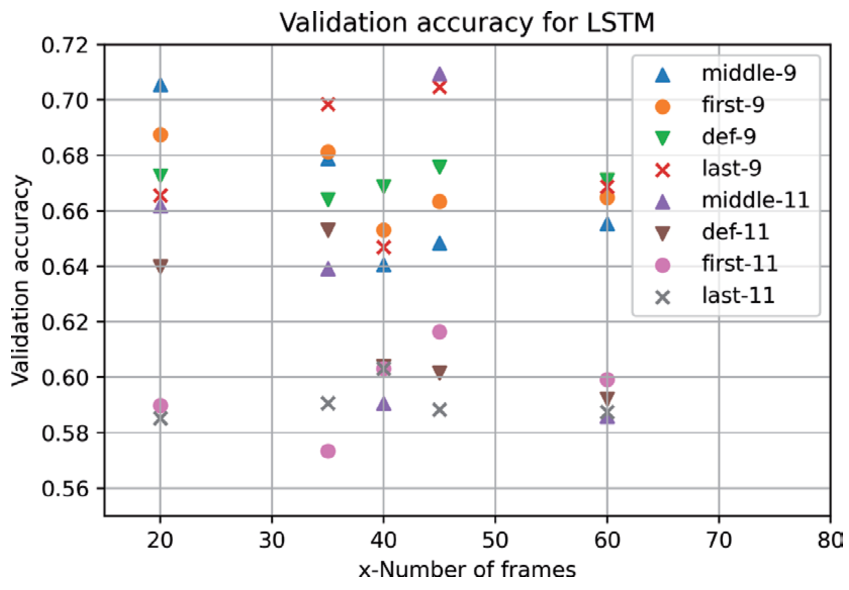


Figure 10.
Validation accuracy for different lengths of input sequences and 9 or 11 action classes.

improve the result much over the models trained with 20 frames. Considering only the way the sequence is selected, the highest average accuracy of 67.69% is achieved by the model while trained on 9 classes and the last frames selection, followed by 67,05% by skipping frames.

It can be noted that regardless the strategy of selecting frames, increasing the number of input frames does not contribute to a better result. The number of frames and frame selection strategies appear to be highly dependent on the type of action being performed.

8. Application of low-level video features

Low-level features extracted from video frames, combined with specific knowledge about the problem domain can sometimes be used for solving specific tasks and generate conclusions about the objects in the image or for scene analyzes. For example, optical flow can be used as a measure of motion in video, and for rough temporal segmentation of the input video in order to automatically cut periods of inactivity or detect intervals of repetition of a certain exercise in handball training.

If the low-level features such as optical flow or spatio-temporal interest points are used with additional information such as the detected player bounding boxes, conclusions can be drowned about the most active player on the scene and automatic detection of players that are at a certain time likely to be performing the action that is of most interest for the interpreting the scene [27].

8.1 Temporal segmentation using optical flow

A low-level feature that captures motion information is optical flow, which is estimated from the time-varying image intensity. A moving point on the image plane produces a 2D path $\boldsymbol{x}(t) \equiv (x(t), y(t))^T$ in camera-centered coordinates, and the current direction of movement is described by the velocity vector $d\boldsymbol{x}(t)/dt$. The 2D velocities of all visible surface points form the 2D motion field.

The movement of points can be estimated from consecutive video frames, using some optical flow estimation algorithm, e.g., the Lucas-Kanade method [28]. This method assumes that small sections, i.e., groups of pixels of the initial images move with the same velocity, so the result is a vector field V of velocities of each image section. At each point $V_{x,y}$ in the field, the vector magnitude corresponds to the speed of movement and the vector angle represents the movement's direction.

A visualization of an optical flow field calculated between two video frames from the dataset is shown in **Figure 11**. The direction and magnitude of optical flow at each point is represented by the direction and length of each arrow.

In an uninterrupted recording of a handball training, there are usually periods of repletion of a certain exercise, where all players repeat the exercise either simultaneously or taking turns, followed by short pauses where the coach explains the next

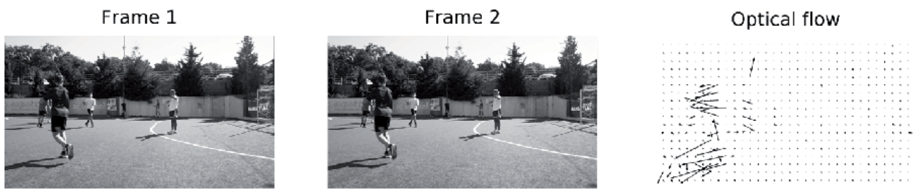


Figure 11.
Two consecutive frames in video and the corresponding optical flow field.

exercise to be performed. The periods of higher activity, when players perform the exercises are characterized with the higher magnitude of extracted motion features from video, while the periods when players queue or wait for instruction (**Figure 12**) are characterized with lower magnitude of motion features [29].

To mark the periods of inactivity and segment the videos into sections where a single exercise is repeatedly practiced, an optical flow threshold is used. First, the optical flow field is calculated between two consecutive frames sampled each N frames (here, N = 50). Then, mean optical flow magnitude is calculated for each field, resulting in a single value for each sampled time point in video. The video is cut at time points when the mean magnitude of optical flow is lower than an experimentally determined threshold value. An example of the mean optical flow magnitude calculated for a video sequence where there were short pauses of 10–20 seconds between active repetition of an exercise was is shown in **Figure 13**. It can be seen that the normalized flow threshold of about 0.07 clearly separates the periods of inactivity from parts of video showing exercise.

8.2 Active player determination

In a typical footage of a handball game or training session, at a given time only one player or a small proportion of players present on the scene participate in the action that is currently in focus, e.g., jump-shot, passing, while others may perform actions that are not currently relevant for interpreting the situation, e.g., moving into their positions. To train the action recognition models, the actions of those players that perform the action of interest should be annotated. The annotation process is at least partly manual, so it is time-consuming and tedious given the large amounts of video data to process. To assist with annotation, a method is proposed to select among the detected and tracked players, the ones that are currently the

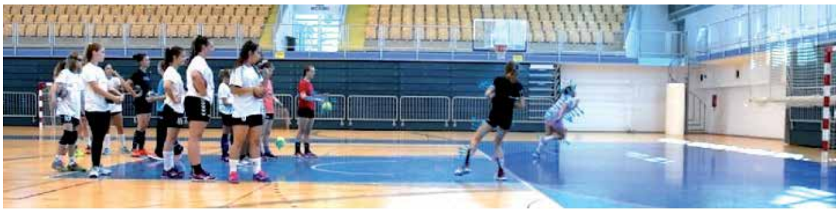


Figure 12.
A typical training situation. Two players on the right are performing the current task, while the rest are queuing.

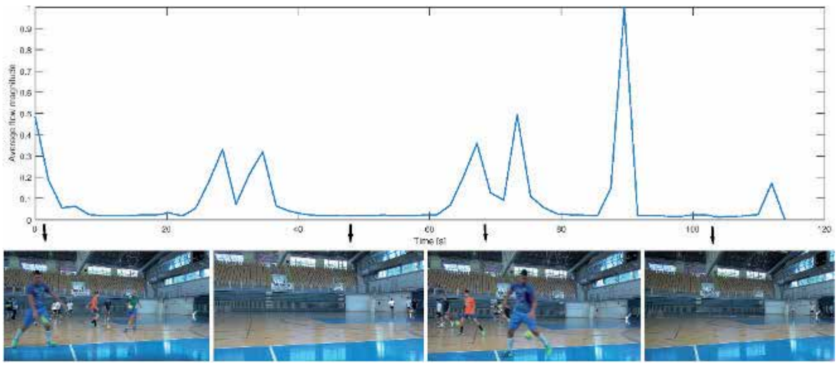


Figure 13.
An example of segmentation of a video sequence using optical flow magnitude.

most likely to be performing the action of interest, here called active players. Thus, instead of reviewing every single players' activity at all times, the manual annotation is reduced to verification of only the proposed active players' tracks.

First, the players are detected and tracked as described in previous chapters. Then, the information about player positions, i.e., the detected bounding boxes, is combined with the low-level movement features, such as optical flow or spatiotemporal interest points, to obtain a measure of each player's activity in the considered time interval.

The optical flow-based measure of player's activity (A_b^{OF}) is calculated using the bounding box (B_b) of each detected player bounding box, as the maximum optical flow magnitude within that box:

$$A_b^{OF} = \max_{B_b} \left| V_{x,y} \right|; x, y \text{ within } B_b \tag{5}$$

An alternative feature to optical flow for defining the activity measure are spatiotemporal interest points, or STIPs. STIPs are an extension from the spatial domain into both spatial and temporal domain of the notion of interest points in images, which are points with a significant local variation of image intensities. STIPs are thus points in the image with large variation of values in both spatial and temporal directions around these points.

As for calculating the optical flow, there are several algorithms that can be used to detect the STIPs, e.g. the method presented in [30] is based on the Harris corner operator (Harris3D) that is extended into the spatiotemporal domain, [31] uses a Gaussian filter in the spatial domain and a Gabor band-pass filter in the temporal domain, the algorithm presented in [32] is based on Hessian3D derived from SURF, and the selective STIPs detector [33] is designed to specifically detect the STIPs that likely belong to persons and not to the background.

Given that movement during the performance of various sports actions causes significant variation of appearance in that part of image, it is expected that there will be more detected STIPs in image regions with more intense player's activity [34].

An activity measure based on density of STIPs in the area near the detected player, A_b^{STIP} , can be calculated for a player with a bounding box B_b and area P_b as:

$$A_b^{STIP} = \frac{\#STIP}{P_b}; STIP \text{ within } B_b. \tag{6}$$

In the experiment here, the Harris3D detector with the default parameters was used to extract the STIPs. **Figure 14** shows an example of the detected player bounding boxes and STIPs in a video frame.

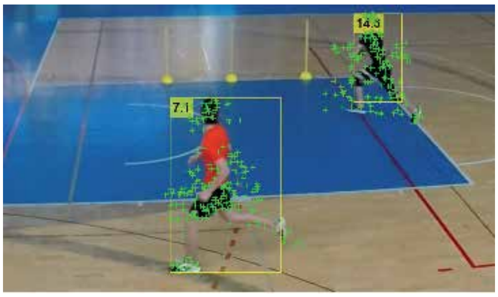


Figure 14.
Detected players and spatiotemporal interest points.

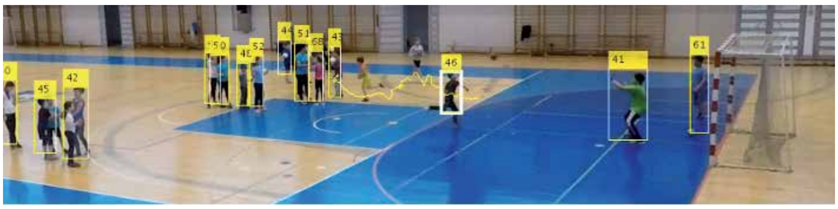


Figure 15.
Detected leading player (white box) and his trajectory through the whole sequence (yellow line).

A threshold of activity measure can be used to filter active from inactive players, since the players that perform sports actions should make more sudden movements corresponding to higher activity measures than other players.

Looking at activity measures in a sequence of frames, the ranking of players' activity can change between frames. So, to choose the players that are active throughout the sequence, the Active player score is calculated as the average activity measure of the player along the trajectory of the player's bounding boxes. The result is a set of player trajectories with corresponding player activity scores (**Figure 15**).

9. Conclusion

In this chapter, the applications of deep learning methods on typical CV tasks such as object detection, object tracking and action recognition are presented on videos from the handball domain, recorded during training and matches.

Handball is a team sport, played with a ball, with well-defined player's roles, goals and rules. During the game, the athletes move quickly throughout the field, change positions and roles from defensive to offensive and vice versa, use different techniques and actions, and doing so often get partially or completely occluded by another athlete, making player detection, tracking and action recognition challenging problems of ongoing research interest.

For detection, the algorithm must be able to locate an object in relation to its environment and, define that object. It is important for the detector to be as accurate and fast as possible especially if the real time detection is needed. State of the art deep learning-based detectors such as YOLOv3 and Mask R-CNN, prove to be successful for player detection, while the performance on ball detection still lags due to the combination of its small size, great speed and occlusion by the players.

Once objects such as players are detected, they can be tracked. Here, the Hungarian assignment algorithm and SORT with a deep association metric (Deep SORT) are considered for tracking. The goal of a tracker is to assign the same unique track ID to the same player in consecutive frames, which is complicated by the changes of appearance and sudden motions of players. Thus, the trackers can model this motion or the changing appearance to help the association process. The Deep SORT adds an appearance model based on deep neural network features. This appearance model allows the Deep SORT method to re-identify players that have been temporarily occluded or left the scene much more successfully than the other tested methods, making it more appropriate for use in the handball domain.

For the action recognition task, LSTM network is used, as it is suited to deal with both image information contained in a single video frame and its temporal evolution during the performance of actions. The obtained action recognition results are promising, however due to dependence of the action recognition model on the performance of previous stages, i.e. object detection and tracking, the challenge remains to improve all three tasks. As in all deep learning tasks, an important factor is gathering enough training data, which can be facilitated by methods that reduce the manual effort of labeling ground truth data. To that end, the experiments for automatic temporal segmentation of the raw footage and a method for detecting the active player in a sequence using low level visual features were presented.

Advances in deep learning methods promise continued improvement in the analysis of dynamic sports scenes, in order to recognize more complex activities, plan competitive tactics and monitor player progress.

Acknowledgements

This research was fully supported by the Croatian Science Foundation under the project IP-2016-2106-8345 "Automatic recognition of actions and activities in multimedia content from the sports domain" (RAASS) and by the University of Rijeka under the project number uniri-drustv-18-222.

Author details

Marina Ivasic-Kos*, Kristina Host and Miran Pobar
Department of Informatics, Center for Artificial Intelligence and Cybersecurity, University of Rijeka, Rijeka, Croatia

*Address all correspondence to: marinai@uniri.hr

References

[1] S. J. D. Prince, Computer Vision: Models, Learning, and Inference, Cambridge Univesity Press, 2012.

[2] M. Nixon and A. Aguado, Feature Extraction and Image Processing for Computer Vision, Elsevier, 2020.

[3] K. Soomro and A. Zamir, "Action Recognition in Realistic Sports Videos," *Springer,* 2014.

[4] P. Pareek and A. Thakkar, *A survey on video-based Human Action Recognition: recent updates, datasets, challenges, and applications,* Springer Nature, Artificial Intelligence Review, 2020.

[5] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," in *International Conference on Engineering and Technology (ICET)*, Antalya, 2017.

[6] M. Buric, M. Pobar and M. Ivasic-Kos, "Adapting YOLO Network for Ball and Player Detection," in *8th International Conference on Pattern Recognition Applications and Methods*, 2019.

[7] K. Host, M. Ivasic-Kos and M. Pobar, "Tracking Handball Players with the DeepSORT Algorithm," in *9th International Conference on Pattern Recognition Applications and Methods*, Valletta, 2019.

[8] J. Johnson and A. Karpathy, "Convolutional Neural Networks," Stanford Computer Science, [Online]. Available: https://cs231n.github.io/convolutional-networks. [Accessed 11 3 2019].

[9] D. M. W. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," *International Journal of Machine Learning Technology,* pp. 37-63, 2011.

[10] E. Ristani, F. Solera, R. S. Zou, R. Cucchiara and C. Tomasi, "Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking," 2016. [Online]. Available: http://arxiv.org/1609.01775.

[11] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[12] A. Farhadi and J. Redmon, "YOLO9000: Better, Faster, Stronger.," in *IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, 2017.

[13] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," Apr 2018. [Online]. Available: https://arxiv.org/1804.02767.

[14] D. Cook, K. D. Feuz and N. C. Krishnan, "Transfer learning for activity recognition: a survey," *Springer London,* 2013.

[15] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," 2017. [Online]. Available: https://arxiv.org/1703.06870.

[16] S. Ren, K.He, R. Girshick and J.Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," 2015. [Online]. Available: https://arxiv.org/1506.01497.

[17] P. L. Mazzeo, P. Spagnolo, *M. Leo* and T. D'Orazio, "Visual Players Detection and Tracking in Soccer Matches," in *IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance*, Santa Fe, 2008.

[18] H. W. Kuhn, "The Hungarian method for the assignment problem," in *Naval Research Logistics Quarterly*, vol. 2, 1955, pp. 83-97.

[19] N. Wojke, A. Bewley and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *International Conference on Image Processing (ICIP)*, Septe, 2017.

[20] M. Buric, M. Ivasic-Kos and M. Pobar, "Player Tracking in Sports Videos," in *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2019.

[21] T. D'Orazio, *M. Leo*, P. Spagnolo, P. Mazzeo, N. Mosca, M. Nitti and A. Distante, "An Investigation Into the Feasibility of Real-Time Soccer Offside Detection From a Multiple Camera System," *IEEE Transactions on Circuits and Systems for Video Technology,* pp. 1804-1818, Dec. 2009.

[22] G. Kayumbi, P. Mazzeo, P. Spagnolo, M. Taj and A. Cavallaro, "Distributed visual sensing for virtual top-view trajectory generation in football videos," in *7th ACM International Conference on Image and Video Retrieval*, Niagara Falls, 2008.

[23] L. Medsker and L. Jain, Reccurent Neural Networks, CRC Press, 2001.

[24] H. Sak, A. Senior and F. Beaufays, "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling," Google, USA.

[25] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *EEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, 2016.

[26] J. Deng, W. Dong, R. Socher, K. L. L. Li and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*, Miami, 2009.

[27] M. Ivasic-Kos and M. Pobar, "Building a labeled dataset for recognition of handball actions using mask R-CNN and STIPS," in *7th IEEE European Workshop on Visual Information Processing (EUVIP)*, Tampere, 2018.

[28] J. Barron, D. Fleet and S. Beauchemin, "Performance of optical flow technique," p. 43-77, 1994.

[29] M. Ivasic-Kos, M. Pobar and J. Gonzàlez, "Active Player Detection in Handball Videos Using Optical Flow and STIPs Based Measures," in *13th International Conference on Signal Processing and Communication Systems (ICSPCS)*, 2019.

[30] I. Laptev, "On space-time interest points," *Int. J. Comput. Vis. 64,* p. 107-123, 2005.

[31] P. Dollár, V. Rabaud, G. Cottrell and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005.

[32] A. Klaser, M. Marszalek and Schmi, "Spatio-Temporal Descriptor Based on 3D-Gradients," in *BMVC 2008-19th British Machine Vision Conference, British Machine Vision Association,* , Leeds, 2008.

[33] B. Chakraborty, M. Holte, T. Moeslund and J. Gonzàlez, "Selective spatio-temporal interest points," *Computer Vision and Image Understanding 116,* p. 396-410, 2012.

[34] M. Pobar and M. Ivasic-Kos, "Detection of the leading player in handball scenes using Mask R-CNN and STIPs," in *Eleventh International Conference on Machine Vision*, 2018.

Chapter 9

Application of Artificial Neural Networks to Chemical and Process Engineering

Fabio Machado Cavalcanti, Camila Emilia Kozonoe, Kelvin André Pacheco and Rita Maria de Brito Alves

Abstract

The accelerated use of Artificial Neural Networks (ANNs) in Chemical and Process Engineering has drawn the attention of scientific and industrial communities, mainly due to the Big Data boom related to the analysis and interpretation of large data volumes required by Industry 4.0. ANNs are well-known nonlinear regression algorithms in the Machine Learning field for classification and prediction and are based on the human brain behavior, which learns tasks from experience through interconnected neurons. This empirical method can widely replace traditional complex phenomenological models based on nonlinear conservation equations, leading to a smaller computational effort – a very peculiar feature for its use in process optimization and control. Thereby, this chapter aims to exhibit several ANN modeling applications to different Chemical and Process Engineering areas, such as thermodynamics, kinetics and catalysis, process analysis and optimization, process safety and control, among others. This review study shows the increasing use of ANNs in the area, helping to understand and to explore process data aspects for future research.

Keywords: chemical and process engineering, Artificial Neural Networks, Big Data, Modeling

1. Introduction

Many chemical and process engineers are excited about the applications of Artificial Intelligence (AI) to their fields of expertise. AI can be defined as the ability of digital-computers to perform tasks at which people are better, at the moment [1]. In this context, Machine Learning (ML) is seen as one of the most relevant subareas, providing computers with the ability to learn without being programmed explicitly. It is essentially a form of applying statistics to estimate complex functions with less emphasis on obtaining the confidence intervals around them [2].

This current excitement was also stimulated by the Big Data boom related to the analysis and interpretation of large data volumes (of the order of several terabytes), which are generated at high rates and present various formats (numbers, text, multimedia, among others). Industry 4.0 requires this piece of knowledge from chemical and process engineers since process plants have large volumes of stored historical data, obtained through sensors that measure thousands of variables in the order of seconds [3]. The analysis and exploitation of these data is a critical component for the operation of an industrial process.

In this framework, the so-called Artificial Neural Networks (ANNs) have numerous advantages and applications. They are universal nonlinear approximators based on the human brain-behavior through interconnected neurons that learn tasks from experience; in that case, from data [4]. Similarly to the nervous system, artificial neural networks are organized in the form of several simple individual elements – nodes or neurons – which interconnect with each other, forming networks capable of storing and transmitting information from/to the outside. Another relevant capacity of ANNs is their plasticity, which, through a learning process, allows changing in the interconnection pattern of its elements [5]. ANNs have been widely used in modeling or regression (linear and nonlinear) for one or several independent variables. It is worth noting that their use is not new in Chemical and Process Engineering, dating from the 1980s with some progress along the way, decisively contributing to the resurgence of the interest of the scientific community in this subject [3].

Modeling, simulation, and optimization are essential activities and competitive differentials among researchers to meet the challenges produced by environmental and commercial restrictions. In this context, ANNs are mostly used in process prediction and classification, as they are a robust nonlinear regression. In particular, this technique should be used when the solution of a problem is hampered by some of the following points: lack of physical or statistical understanding of the problem, statistical variations of the observable data, and the nonlinear mechanism responsible for the generation of the data [6].

In general terms, the use of neural networks consists of the following steps: 1- establishing the network architecture; 2- providing experimental data; 3- adjusting the network parameters – also known as their weights – until they learn the phenomenon (step called training); and 4- using the trained network with new input data for predicting the corresponding output data.

ANNs have been successfully applied to chemistry to correlate spectra of analytical methods and product properties [7]; in catalysis, to determine the relationships between the catalyst structure and its activity [8]; in process modeling, to predict product performance and operating conditions [9], and particularly in process control and fault diagnosis [10]. The main reasons for the growing popularity of the neural network approach are its lower computational cost compared to other methods and its ability to solve complex nonlinear problems [5]. Therefore, this review study demonstrates the increasing use of ANNs in Chemical and Process Engineering, helping to understand and to explore process data aspects for future research.

2. Most common activation functions used in chemical and process engineering applications

A neural network contains hyperparameters to be tuned prior to training in order to achieve the best configuration. Among them, the following can be mentioned: (i) number of hidden neurons, (ii) activation function, (iii) optimizer, and (iv) regularization and their dependencies (learning rate, optimizer specific, dropout rate, etc.).

Particularly, activation functions determine the output of the model, its accuracy, and the computational efficiency of training a model; therefore, they are an essential part of the structure of the neural networks. The Sigmoid function, Hyperbolic Tangent (TanH), and ReLU (Rectified Linear Unit) are the most common in Chemical Engineering; however, recent studies improve these classical activation functions, defining new ones, such as Leaky ReLU, Swish, H-Swish [11].

In the sigmoid activation function, the output values are bounded between 0 and 1, normalizing each neuron output. However, there is a problem with the vanishing gradient, and outputs are not zero-centered. To make the modeling easier, the TanH was proposed, for which the outputs are zero-centered, i.e., when the inputs contain strongly negative, neutral, and strongly positive values.

In order to circumvent the computational expense, the ReLU was proposed. It is a computationally efficient linear activation function that will output the input directly if it is positive; otherwise, it will output zero. A further development is the Leaky ReLU, whereby the slope is changed to the left of $x = 0$, avoiding the dying ReLU problem, whereby some neurons can die for all inputs and remain inactive.

Therefore, the correct definition of the activation function is a fundamental part of the hyperparameter tuning to guarantee the best configuration of a neural network. In the course of the chapter, we will always mention which activation function each work used in the summary tables.

3. Applications to chemical and process engineering

In recent decades, there have been a large number of studies using ANNs in chemical engineering, from molecular property prediction [12], fault diagnosis [13], predictive control [14], and optimization [15, 16]. The use of first-principles knowledge must be integrated with the neural network in order to retain a more physical understanding of the system [14]. In the following subsections, we presented the principal papers of each area, with tables summarizing the characteristics of the ANNs used.

3.1 Thermodynamics and transport phenomena

Several data-driven models have been employed to predict phase equilibrium and transport phenomena coefficients for various chemical systems [17]. Indeed, these fields already have some empiricism in their standard mathematical formulations. For example, flash algorithms have some empiricism when using binary interaction parameters in subjective mixing rules [18], and the majority of transport phenomena coefficients are estimated from empirical correlations, sometimes questionable [19]. Therefore, the use of ANNs is a better way to find functional relationships between the model variables instead of first determining these constants [20].

Moreover, ANNs reveal a conceivably faster choice to those property prediction calculations in process simulations, limiting process control applications that require to be conducted in real-time. For this, Poort et al. [21] studied the replacement of conventional Equations of State (EoS) for property and phase stability calculations on a binary mixture of methanol–water. They trained ANNs with data generated through the Thermodynamics for Engineering Applications (TEA) to represent four kinds of flash algorithms, leading to an enhancement of 15 times for the predictions of properties and 35 times for classification of the phases.

Also noteworthy is that ANNs have also been used to predict if a particular mixture forms an azeotrope – essential information to design and to control a separation process. Alves et al. [22] successfully developed an ANN classification model to determine whether binary mixtures can exhibit (or not) azeotropy based solely on the properties of pure components as input variables. Therefore, it shows the power of ANNs for this type of thermodynamic evaluation since it does not take into account the non-ideality of the mixture.

They are also widely employed to predict thermal-physical properties of ionic liquids, such as density and viscosity [23]. The primary source of these values comes from experiments at the laboratory since ionic liquids do not present a universal description of their phase behavior. For example, using the definition of group contribution and the operating temperature, Valderrama et al. [24] successfully developed a three-layer FF-ANN to estimate the density of ionic liquids.

ANNs have also been employed in statistical thermodynamics techniques, which compute physicochemical properties from molecular simulations. One of these methods – the High-Throughput Force Field Simulation (HT-FFS) – can generate large volumes of data. ANNs can be trained with these data, thus building a gray-box model to improve the property predictions with a lower computational effort [25]. They have also been used in Density Functional Theory (DFT) calculations to replace some physical functionals with data-driven ones, finding the energy levels for electronic structures of different compounds with a balance between computational cost and accuracy [26].

Regarding their application to Transport Phenomena, it is well-known that ANNs – as an excellent universal approximator for any nonlinear function [27] – can be used for estimating convective heat- and mass-transfer coefficients [17]. Mainly in situations in which there is no mathematical correlation that can adjust them, as is the case of bubble columns. For this, Verma and Srivastava [19] successfully built an ANN model from literature data with eight inputs related to the system configuration of a bubble column (gas velocity, Prandtl number, number of holes, hole diameter, column diameter, surface tension, gas holdup, and bed height) and one output (heat coefficient).

Table 1 displays a summary of the current applications of neural networks to thermodynamics and transport phenomena discussed above. In the table, we specify the field, case study, class of neural network, activation function, topology and software used in each work.

3.2 Kinetics and catalysis

Neural networks have been successfully applied to catalysis to determine the relationship between the catalyst structure and its activity [8]. As heterogeneous catalysis has developed increasingly efficient experimentation techniques, the number of new data have increased exponentially [28], both from synthesis and from characterization and catalytic tests [29]. Thus, there is a need for more adequate tools to manage these large amounts of experimental data, to understand and to model it, and to generate a way to optimize the catalytic performance [30].

Two types of ANNs applications have been described so far in the frame of combinatorial catalysis: (i) ANN catalyst compositional models, correlating composition and synthesis variables with catalytic performance, and (ii) ANN kinetic models, correlating reaction conditions with the catalytic performance [31]. For example, those applications include the design of ammoxidation of propylene catalyst [32], design of methane oxidative decoupling catalyst [33], analysis and prediction of results of the decomposition of NO over zeolites [34], among other studies. Also, ANNs have been used combined with genetic algorithms for designing propane ammoxidation catalysts [35]. Another work successfully reported the viability of ANNs in the analysis and prediction of catalytic results within a collection of catalysts produced by combinatorial techniques [36]. Recently, an ANN was applied to estimate the rate of dehydration reaction of methanol in dimethyl ether synthesis [37]. The results showed that an ANN is a powerful tool for evaluating the reaction rate instead of using sophisticated kinetic model equations.

References	Field	Case Study	Class of Neural Network	Activation Function	Topology***	Software
[18]	Phase Equilibrium	Vapor–Liquid equilibrium of NH_3/H_2O and CH_4/C_2H_6 systems	FF-ANN*	Sigmoid	2–13-2	in-house software
[20]	Transport Phenomena	Determination of reduced boiling point from molecular weight and acentric factor	FF-ANN	Sigmoid	2–2–2-1	Matlab
[21]	Phase Equilibrium	Vapor–liquid flash calculations	FF-ANN	Linear/ Sigmoid	3–10-2	Keras-Python
[22]	Phase Equilibrium	Prediction of azeotrope formation	FF-ANN	Sigmoid	16–6-1	in-house software
[24]	Ionic Liquids	Estimation of physical properties of ionic liquids	FF-ANN	Tanh	10–15–15-1	Matlab
[25]	Molecular Thermodynamics	Enhancing the High-Throughput Force Field Simulation (HT-FFS)	FF-ANN	Linear/ ELU**	25–16–8-4-3	PyTorch
[26]	Molecular Thermodynamics	Correlation functionals of the electronic density	Fully connected neural networks	Sigmoid	4–8 neurons in each hidden layer	TensorFlow

**FN-ANN stands for Feed-Forward Artificial Neural Network.*
***ELU stands for Exponential Linear Unit.*
****The first and last elements in topology represent the number of neurons in the input and in the output layer, respectively. Among them, the number of neurons in the hidden layer(s).*

Table 1.
Current applications of ANNs to thermodynamics and transport phenomena.

The number of publications in this catalysis field has had an upward trend, especially in the last decade with the high demand for practical applications of the concepts of Big Data. The group of Turkish researchers led by Günay and Yildirim has excelled with work in the field, using not only ANNs for extracting knowledge from catalytic data, but also decision tree algorithms to determine the heuristic conditions and rules that lead to a high performance of the catalyst. For example, in work about carbon monoxide oxidation over Cu-based catalysts, they successfully used 1337 data points from 20 studies for evaluating catalyst performance using ANNs [38].

In the field of heterogeneous catalysis, ANNs can be used to select better possible catalysts – cheaper, less toxic, and composed of non-precious metals – for a given reaction, thus reducing the massive number of needed high-throughput experiments, peculiar conjuncture of combinatorial catalysis [39]. In this direction, Cavalcanti et al. [40] used a three-layer feedforward neural network to predict the ideal composition of the catalyst in the water-gas-shift reaction and discover useful trends through sensitivity analysis. The input variables for ANN were several, while the only output variable considered was the conversion of CO. The model for the reaction was successfully developed, exhibiting the power of ANNs for predicting better catalysts and operating conditions for the process.

Recently, Cavalcanti et al. [8] showed that ANNs are able to predict the variables that most influence the conversion of CO in the water-gas-shift reaction, that is, temperature and surface area. The results can be used to conduct subsequent research in an optimized manner in this area, as it aims at the well-managed use of environmental resources, in the sense of selecting efficient catalysts for producing hydrogen - a clean energy source.

In the same topic, Garona et al. [41] presented an empiric model for the Fischer-Tropsch Synthesis (FTS) reaction using ANNs. A database of FTS to light olefins was assembled from the literature, and feedforward neural networks were used to build more complete models, which helped to predict optimal catalyst composition and operating conditions.

It is also noteworthy that ANNs were also used to model the sintering of a catalyst in a dry reformer [42]. In particular, the effects of temperature, pressure, and catalyst diameter on methane and CO_2 conversions, H_2/CO ratio, and molar percentage of solid carbon deposited on the catalyst (responsible for deactivation) have been studied. The ANN design activity was automated using a Genetic Algorithm (GA) search over the set of possible network topologies. The inclusion of the effective number of parameters in the GA objective function led to networks that performed well over testing data points.

Another application is in the determination of acidity in zeolites with data from FTIR spectroscopy [43]. FF-ANNs were used for analyzing multivariate base on the characteristic absorbance of 11 zeolite samples after metal substitution (Zn, Cu, Ga, and Ag) in the ~3612 cm^{-1} region. The developed regression method presented the same results of acid sites from other conventional and expensive methodologies.

Thus, in order to formulate a new kind of catalyst, it is essential to identify the catalysis past [44]. Therefore, by using ANNs, it is possible to convert historical data from past publications into valuable information, leading to a great acceleration in the development of new catalysts with better performances for a given process [8]. **Table 2** presents a summary of the current applications of neural networks to catalytic processes.

3.3 Process analysis and optimization

The applications of neural networks to the process analysis are increasing. Assidjo et al. [45] modeled the drying process of the production of coconut using a neural network. The goal is to predict the moisture of dried grated coconut whose dynamics are not well known. The authors used a feedforward fully connected neural network, whereby the selected architecture was 9–4-1, selected based on the minimum error in the test set. The results indicate that the neural network proposed, constructed using industrial plant data, can be used as a predicting method.

Fernandes and Lona [46] applied neural networks to the field of polymerization. The authors also highlighted some topologies, the number of data points needed, and the concept of stacked neural networks that can enhance the prediction of the final model.

Alves and Nascimento [47] used industrial plant data for constructing neural networks to detect gross errors; the case study was an isoprene unit facility.

Alves and Nascimento [4] studied the production of high purity isoprene from a C_5 cut arising from a pyrolysis gasoline unit. The first principle models were replaced by neural networks in the final grid search of the optimal parameters for the process. The set of 10 neural networks were defined to represent the whole flowsheeting, whereby the number of hidden layers was defined by the minimum error in the test set. Lastly, the framework successfully optimized a chemical plant under study using neural networks with industrial data.

References	Field	Case Study	Class of Neural Network	Activation Function	Topology**	Software
[29]	Modeling of catalytic processes	Catalytic activity for n-paraffin isomerization	FF-ANN*	Sigmoid/ Tanh	4–8–6-3	SNNS neural networks simulator
[32]	Catalyst design	Design of catalyst for propane ammoxidation	FF-ANN	Sigmoid	6–20–12–2	in-house software
[33]	Catalyst design	Design of a catalyst for methane oxidative coupling	FF-ANN	Sigmoid	6–20–9-2	in-house software
[34]	Modeling of catalytic processes	Analysis of NO decomposition over Cu/ZSM-5 zeolite	FF-ANN	Sigmoid	4–32-1	in-house software
[36]	Combinatorial catalysis	Modeling of catalysts for oxidative dehydrogenation of ethane	FF-ANN	Not described	13–26–12-6	SNNS neural networks simulator
[37]	Modeling of catalytic processes	Estimation of the reaction rate in methanol dehydration	FF-ANN	Tanh/ Linear	3–6-1	Matlab
[38]	Modeling of catalytic processes	Selective CO Oxidation over Copper-Based Catalysts	FF-ANN	Tanh	14–7-7-1	Matlab
[8]	Catalyst selection	Catalyst selection for the WGS reaction	FF-ANN	Sigmoid	51–12-1	R - neuralnet
[41]	Modeling of catalytic processes	Fischer-Tropsch synthesis to lower-olefins	FF-ANN	Sigmoid	30–15-2	R - neuralnet
[42]	Catalyst deactivation	Dry reformer under catalyst sintering	FF-ANN	Tanh	3–12–5-6-1	in-house software
[43]	Determination of catalyst acidity	Determination of acidity in metal incorporated zeolites by FTRI	FF-ANN	Tanh	6–10-1	Matlab

**FN-ANN stands for Feed-Forward Artificial Neural Network.*
***The first and last elements in topology represent the number of neurons in the input and in the output layer, respectively. Among them, the number of neurons in the hidden layer(s).*

Table 2.
Current applications of ANNs to catalytic processes.

Khezri et al. [15] proposed a hybrid model for optimizing a large-scale gas to liquids process. The dataset was constructed using a simulation model of the GTL process. Different topologies were compared to select the most promising one; one and two hidden layers with different number of neurons were tested. The optimal configuration was two hidden layers with 7 and 15 hidden neurons each. The ANN was modeled using the information of the tail gas unpurged ratio, recycled tail gas to FT ratio, H_2O/C in the syngas section, and CO_2 removal percentage as input features; the outputting was wax production rate. The ANN model was then used for optimization purposes.

Wang et al. [16] proposed a framework for predicting the operating trend of an industrial process. The framework contains three major steps: (i) multivariate correlation analysis, to deal with the correlation between the historical industrial data, (ii) clustering, due to nonlinear dense data and unclear operating trend types and (iii) a convolutional neural network (CNN), formed by five parts (input layer, convolutional layer, ReLU layer, pooling layer, and fully connected layer).

The authors pointed out the importance of the convolutional networks to extract important features from the dataset. Moreover, the advantage of such a framework was compared with traditional convolutional neural networks and recurrent neural networks (RNNs) for a methanol production process.

Cai et al. [48] analyzed an industrial process using data-driven models. The case study was the industrial reverse osmosis concentrate (ROC) treatment with the fluidized bed reactor Fenton (FBR-Fenton) process. Prior to modeling, a statistical analysis was carried out to determine the most relevant features as input (Fe^{2+} dosage, H_2O_2 dosage, pH, and HRT). Two approaches were studied, ANN and linear regression. The former showed more accurate predictions, consisting in one input layer (4 neurons), 4 hidden layers (10 neurons each) and one output layer (2 neurons) using ReLU as an activation function, due to the least computationally dense mechanism and also a general approximation for most scenarios [11].

The crystallization process and the quality of the products was studied by Lin et al. [49]. The authors used a Raman spectrum as input for a two-layer back propagation neural network with four hidden neurons to predict the solution concentration and slurry density simultaneously. They also compared the output prediction of the neural network with other algorithm predictions (characteristic peaks regression, principal component regression, partial least-squares regression), and the results indicated the superior prediction characteristics of the neural network due to its inner nonlinear nature.

Chemical process synthesis is a complex scheme, which comprises process modeling and design, and combinatorial defiance. There are two major approaches: the traditional sequential form and the optimization-based synthesis using superstructure models. In the former category, the problem is solved in sequential scheme, by decomposition whereby there is a hierarchy of elements that can be depicted by an Onion Diagram (reactor, separation, heat recovery and utility) [50].

The latter category considers the full integration between decisions at the single step, i.e. determine the optimal structure and operating conditions simultaneously. Therefore, this approach contemplates all possible complex interactions between the engineering choices, including equipment (potentially selected in the optimized flowsheet), the interconnection and operating conditions formulated as an optimization problem [51–53].

There is a diversity of proposed methodologies to represent a general process superstructure [54–56]. However, due to the inner complexity of the superstructure (**Figure 1**), the large-scale non-convex Mixed-Integer Nonlinear Programs (MINLP) require effective approaches to solve them.

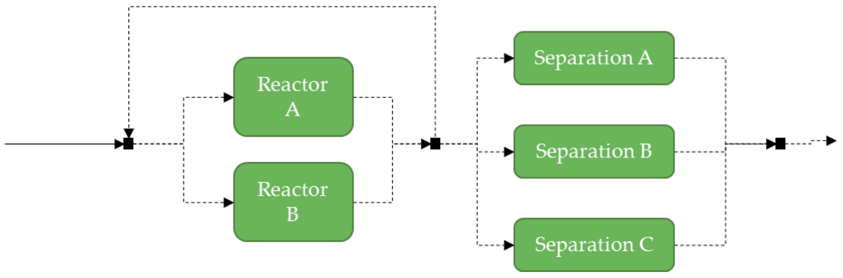


Figure 1.
Simple superstructure representation compared with different separation processes.

The use of simplified models or surrogates at the unit operation level is advantageous because they are present in any process simulator. Additionally, surrogates can be used to represent an entire subsystem consisting of a definite number of units. Artificial Neural Networks (ANNs) may be used to generate the surrogate models, due to their fitting characteristics [57].

In order to circumvent the solution problem of a superstructure, Henao and Maravelias [58] proposed a framework to replace complex unit models (based on first-principle) with surrogate models, developed using artificial neural networks. The authors proposed simpler surrogate models for pumps, compressors and flash vessels. The authors used two case studies (Absorption-based CO_2 capture system and maleic anhydride process superstructure) to validate the proposed framework. The results indicate the possibility of using neural networks embedded in a rigorous optimization procedure.

Savage et al. [59] proposed a hybrid machine learning-based framework to optimize the chemical process (the CryoMan Cascade cycle system was used as a case study). The authors compared different surrogate models algorithms (ANN and Kriging Partial Least Squares); the results indicated a reduction in the time needed for the optimization when compared with the rigorous model. Moreover, they found that a single large ANN was unable to capture the high nonlinearity of the process under study based on the final accuracy. Therefore, the authors broke the surrogate model into a series of parallel sub-models, revealing to have increased the final accuracy.

According to Klemes et al. [60], despite the substantial level of maturity of the process modeling, the nature of connections of the problem still allows improvements. Nascimento et al. [61] also presented alternatives for the optimization of industrial facilities using neural networks and compared them with industrial data.

Table 3 presents a summary of the current applications of neural networks to process analysis and optimization.

3.4 Process safety and control

One of the most common applications of ANNs to the area of process safety and control is in fault detection and diagnosis. These systems are built to identify habitual process behavior and recognize atypical variations in the chemical plant that can lead to an accident [64]. Generally, deep neural networks – ANNs that contain several hidden layers – are used to extract spatial and temporal aspects of the data for this purpose [65]. Their inputs are the sensors responsible for the variable measurement, and their outputs of the kind of faults (e.g., tube plugging, valve blockage, catalyst deactivation, among others) [66].

References	Field	Case Study	Class of Neural Network	Activation Function	Topology****	Software
[45]	Process Analysis	Grated coconut industry	FF-ANN*	Tanh	9–4-1	Matlab
[16]	Industrial Process Operating (Predictive Control)	Methanol production	CNN**	ReLu	5 convolution layers, 36 filters, and 3 pooling layers	Caffe
[62]	Process Analysis	Fluidized bed reactor Fenton process	FF-ANN	ReLu	4–10– 10 – 10 – 10 – 2	R - Keras
[14]	Predictive Control	non-isothermal continuous stirred tank reactors	RNN***	Tanh	2 hidden layers with 30 neurons in each layer	Python-Keras
[15]	Process Optimization	Large scale gas to liquids process	FF-ANN	Sigmoid	4–7–15-1	Matlab
[4]	Process Optimization	Isoprene Process	FF-ANN	Sigmoid	10 neural networks (all with one hidden layer)	in-house software
[58]	Process Synthesis	Absorption-based CO_2 capture and Maleic Anhydride process	FF-ANN	Tanh	Several neural networks (all with one hidden layer)	Matlab
[59]	Process Synthesis	CryoMan Cascade cycle system	FF-ANN	Not Described	Not Described	Python-PyTorch
[63]	Process Analysis	Thermo-catalytic methane decomposition	FF-ANN	Sigmoid	6–9-1	Matlab
[49]	Process Analysis	Crystallization process	FF-ANN	Not Described	two-layer neural network with four hidden neurons	Matlab

**FN-ANN stands for Feed-Forward Artificial Neural Network.*
***CNN stands for Convolutional Neural Network.*
****RNN stands for Recurrent Neural Network.*
*****The first and last elements in topology represent the number of neurons in the input and in the output layer, respectively. Among them, the number of neurons in the hidden layer(s).*

Table 3.
Current applications of neural networks to process analysis and optimization.

However, determining the various hyperparameters of deep neural networks demands a considerable amount of time, which is not suitable for fast online process applications. Based on this, Peng et al. [67] applied a method to reduce the training time of these complex types of network architecture: the Broad Learning

System (BLS). It uses an incremental learning procedure and enlarges the network in width, making a quick training stage possible. They successfully employed this strategy in a batch fermentation process for fault detection utilizing the Affinity Propagation (AP) algorithm in a Long Short-Term Memory (LSTM) deep neural network to cluster distinct stage data.

Another use is in developing models to control the process quality through variables that do not have online sensors. On the one hand, variables such as pressure, temperature, and mass flow rate can be easily measured by manometers, thermocouples, and mass flow controllers, respectively. On the other hand, the online measurement of a variable such as pH in the process is a challenge since no large-scale equipment exists for this, depending on an offline laboratory analysis. Therefore, ANNs can be used to develop these so-called *soft-sensors* to predict quality parameters from a large volume of industrial data, improving the process control quality [68].

Finally, ANNs are also used to replace complex phenomenological models in Model Predictive Control (MPC) architectures and Real-Time Optimization (RTO) strategies [69]. Both applications depend on the model accuracy and the velocity of solving the model equations to drive the controlled variable to the desired set-point. The former is related to dynamic processes and the latter to steady-state operations [69]. Since ANNs have a lower computational response than first-principle models, they are a suitable alternative to make these control strategies possible and efficient.

A successful application of this kind of substitution can be found elsewhere [70], in which an ANN is used to replace a very detailed computational fluid dynamic (CFD) model that represents the synthesis of phthalic anhydride in a fixed-bed catalytic reactor for an MPC structure. Moreover, a hybrid model approach (first-principles combined with ANN) was employed in an MPC by Zhang et al. [69] to drive a reaction process in a continuous stirred tank reactor (CSTR) to optimal operating conditions. They represented the reaction rates by neural networks instead of using the nonlinear Arrhenius Law to describe the reaction phenomenon. Indeed, this well-known equation was used to generate the dataset for training the network under numerous variations in temperature and reactant concentrations. The MPC acted to stabilize the chemical process, driving it to the lowest total cost conditions.

Wu et al. [14] proposed a hybrid machine-learning model that incorporates first principles into a recurrent neural network. The authors studied two models, a partially-connected RNN model and a weight-constrained RNN model and applied them to a chemical process containing two well-mixed, non- isothermal continuous stirred tank reactors in series. The two proposed models outperformed a Lyapunov-based model predictive controller based on prediction accuracy, smoother state trajectories and economic advantages.

It is worth mentioning that ANNs are being used to build detectors to prevent cyber-attacks against process plants [71]. Nowadays, with highly automated systems for controlling chemical plants with real-time operation, breaches in cyber-secure failures can exist, which may cause accidents and economic losses. With this in mind, Chen et al. [71] developed a feedback-MPC control architecture with an ANN-detector that can identify the probabilities of cyber-attacks in networked sensors. Therefore, the applicability of ANNs in these safety and control strategies is very significant for the integrability of industrial plants.

Table 4 shows a summary of the current applications of neural networks to the area of process safety and control.

References	Field	Case Study	Class of Neural Network	Activation Function	Topology****	Software
[67]	Fault Detection	Penicillin fermentation process	LSTM***	Sigmoid	10–20–15-2	Matlab
[68]	Soft Sensors	pH control in a chemical process	RNN**	Tanh	5–14–1	Not described
[69]	Surrogate model in MPC and RTO	Reaction process in a CSTR	FF-ANN*	Tanh	3–10-1	Matlab
[70]	Surrogate model from CFD in MPC	Phthalic anhydride synthesis in a fixed-bed catalytic reactor	RNN	ReLu	3–64–64–1	Keras
[14]	Hybrid model in a MPC	Two-consecutive CSTRs	RNN	Tanh	2–30–30-4	IPOPT-Python
[71]	Cyber Security	MPC integrated with cyber-secure feedback controller	FF-ANN	Tanh	4–12–10-9	Matlab

**FN-ANN stands for Feed-Forward Artificial Neural Network.*
***RNN stands for Recurrent Neural Network.*
****LSTM stands for Long Short-Term Memory.*
*****The first and last elements in topology represent the number of neurons in the input and in the output layer, respectively. Among them, the number of neurons in the hidden layer(s).*

Table 4.
Current applications of ANNs to process safety and control.

4. Future works

Today, ANNs are one of the most found subjects in the scientific literature of Chemical and Process Engineering; and their use tends to continue growing. This can be explained by the launch of Industry 4.0, in which these data-driven models play an essential role in the implementation of some type of intelligent systems in processes [72]. Thus, to remain relevant in this current scenario, companies need specialized professionals on their team. For this reason, this topic has been introduced into the curriculum of most Chemical Engineering degree programs [73].

Indeed, the continuous availability of large volumes of stored data in industrial processes will lead to the development of new ANN approaches for process modeling and data interpretation. These models will deliver more direct relationships between cause and effect variables for process optimization and control through MPC strategies. Therefore, the automation of entire plant units will conduct to intelligent processes, capable of making decisions for safer operation, and with a reliable protection system against cyber-attacks.

Another subarea worth mentioning for future developments is the design of new materials. The use of ANNs has led to a decrease in the number of lengthy and costly laboratory experiments for analyzing the performance of polymers, ceramics, glasses, and mainly, catalysts. Therefore, it is possible to convert data from past publications and from high-throughput (HT) experiments into information, leading to a surprising acceleration in developing new materials with better performances for a given process.

5. Conclusions and perspectives

This chapter presented the ANNs and their Chemical and Process Engineering applications, showing how they have become a powerful tool for modeling chemical processes. This analysis also showed their increasing application, helping to understand and analyze process data features for future research in thermodynamics, transport phenomena, kinetics and catalysis, process analysis and optimization, and process safety and control.

The prospective availability of large volumes of data with good quality will make ANNs one of the most used methods to represent a process, estimate thermodynamic properties, develop new catalysts, replace complex phenomenological models, and improve control and safety strategies. Moreover, in real chemical processes, a particular part of the inputs affect only a section of the outputs. Therefore, the knowledge of first principles embedded in a data driven machine learning model is a challenge for the next studies.

Acknowledgements

We gratefully acknowledge the support of the RCGI – Research Centre for Gas Innovation, hosted by the Universidade de São Paulo (USP) and sponsored by FAPESP – The São Paulo Research Foundation (2014/50279-4) and Shell Brasil. In addition, the authors acknowledge the financial support provided by FAPESP for doctoral scholarships (Grant 2017/11940-5 and 2017/26683-8).

Conflict of interest

The authors declare no conflict of interest.

Author details

Fabio Machado Cavalcanti, Camila Emilia Kozonoe, Kelvin André Pacheco and Rita Maria de Brito Alves*
Escola Politécnica - Universidade de São Paulo, São Paulo, Brazil

*Address all correspondence to: rmbalves@usp.br

References

[1] Rich E. Artificial intelligence. 1st ed. McGraw-Hill, Inc.; 1983.

[2] Goodfellow I, Bengio Y, Courville A, Bengio Y. Deep learning. Vol. 1. MIT press Cambridge; 2016.

[3] Venkatasubramanian V. The promise of artificial intelligence in chemical engineering: Is it here, finally? AIChE J. 2019;65(2):466-478.

[4] Alves RMB, Nascimento CAO. Neural network based approach applied to for modeling and optimization an industrial isoprene unit production. AIChE Annu Meet Conf Proc. 2004;7663-7682.

[5] Himmelblau DM. Accounts of experiences in the aPplication of artificial neural networks in chemical engineering. Ind Eng Chem Res. 2008;47(16):5782-5796.

[6] Haykin S. Redes Neurais - Principios e Praticas. 2nd ed. Bookman; 2001.

[7] Gemperline PJ, Long JR, Gregoriou VG. Nonlinear Multivariate Calibration Using Principal Components Regression and Artificial Neural Networks. Vol. 63, Bricout, J.; Fontes, J. C. Ann. Falslf. Expert. Chlm. 1991.

[8] Cavalcanti FM, Schmal M, Giudici R, Brito Alves RM. A catalyst selection method for hydrogen production through Water-Gas Shift Reaction using artificial neural networks. J Environ Manage. 2019 May; 237:585-594

[9] Nascimento CAO, Giudici R. Neural network based approach for optimisation applied to an industrial nylon-6,6 polymerisation process. Comput Chem Eng. 1998;22:595-600.

[10] Guo L, Kang J. A hybrid process monitoring and fault diagnosis approach for chemical plants. Int J Chem Eng. 2015;2015.

[11] Nwankpa C, Ijomah W, Gachagan A, Marshall S. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. arXiv Prepr. 2018;arXiv:1811.

[12] Hirschfeld L, Swanson K, Yang K, Barzilay R, Coley CW. Uncertainty Quantification Using Neural Networks for Molecular Property Prediction. J Chem Inf Model. 2020 Aug;60(8):3770-3780.

[13] Zhang S, Bi K, Qiu T. Bidirectional Recurrent Neural Network-Based Chemical Process Fault Diagnosis. Ind Eng Chem Res. 2020 Jan;59(2):824-834.

[14] Wu Z, Rincon D, Christofides PD. Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes. J Process Control. 2020;89:74-84.

[15] Khezri V, Yasari E, Panahi M, Khosravi A. Hybrid Artificial Neural Network–Genetic Algorithm-Based Technique to Optimize a Steady-State Gas-to-Liquids Plant. Ind Eng Chem Res. 2020 May;59(18):8674-8687.

[16] Wang Y, Ren YM, Li H. Symbolic Multivariable Hierarchical Clustering Based Convolutional Neural Networks with Applications in Industrial Process Operating Trend Predictions. Ind Eng Chem Res. 2020 Aug;59(34):15133-15145.

[17] Verma AK. Process Modelling and Simulation in Chemical, Biochemical and Environmental Engineering. CRC Press; 2015.

[18] Vashishtha M. Application of artificial neural networks in prediction of vapour liquid equilibrium data.

Proc - 25th Eur Conf Model Simulation, ECMS 2011. 2011;6(Cd):142-5.

[19] Verma AK, Srivastava A. ANN based Model for Heat Transfer from Immersed Tubes in a Bubble Column: Effects of Immersed Surface and Sparger Geometry Conference on Thermal Systems. In: Proceedings of Fourth National Seminar on Thermal Systems. 2003. p. 135-9.

[20] Joss L, Müller EA. Machine Learning for Fluid Property Correlations: Classroom Examples with MATLAB. J Chem Educ. 2019;96(4):697-703.

[21] Poort JP, Ramdin M, van Kranendonk J, Vlugt TJH. Solving vapor-liquid flash problems using artificial neural networks. Fluid Phase Equilib. 2019;490:39-47.

[22] Brito Alves RM, Quina FH, Oller Nascimento CA. New approach for the prediction of azeotropy in binary systems. Comput Chem Eng. 2003;27(12):1755-1759.

[23] Yusuf F, Olayiwola T, Afagwu C. Application of Artificial Intelligence-based predictive methods in Ionic liquid studies: A review. Fluid Phase Equilib. 2021 Mar 1;531:112898.

[24] Valderrama JO, Reátegui A, Rojas RE. Density of ionic liquids using group contribution and artificial neural networks. Ind Eng Chem Res [Internet]. 2009 Mar 18 [cited 2021 Feb 9];48(6):3254-9. Available from: https://pubs.acs.org/sharingguidelines

[25] Gong Z, Wu Y, Wu L, Sun H. Predicting Thermodynamic Properties of Alkanes by High-Throughput Force Field Simulation and Machine Learning. J Chem Inf Model. 2018;58(12):2502-2516.

[26] Dick S, Fernandez-Serra M. Machine learning accurate exchange and correlation functionals of the electronic density. Nat Commun. 2020;11(1).

[27] Hornik K, Stinchcombe M, White H. Multilayer Feedforward Networks are Universal Approximators. Neural Networks. 1989;2(5):359-366.

[28] Erdem Günay M, Yıldırım R. Recent advances in knowledge discovery for heterogeneous catalysis using machine learning. Catal Rev - Sci Eng. 2020;

[29] Serra JM, Corma A, Chica A, Argente E, Botti V. Can artificial neural networks help the experimentation in catalysis? Catal Today. 2003;81(3):393-403.

[30] Senkan S. Combinatorial heterogeneous catalysis - A new path in an old field. Vol. 40, Angewandte Chemie - International Edition. John Wiley & Sons, Ltd; 2001. p. 312-29.

[31] Hattori T, Kito S. Neural network as a tool for catalyst development. Catal Today. 1995 Apr 7;23(4):347-355.

[32] Hou ZY, Dai Q, Wu XQ, Chen GT. Artificial neural network aided design of catalyst for propane ammoxidation. Appl Catal A Gen. 1997 Nov 4;161(1-2):183-190.

[33] Huang K, Chen FQ, Lü DW. Artificial neural network-aided design of a multi-component catalyst for methane oxidative coupling. Appl Catal A Gen. 2001 Oct 5;219(1-2):61-68.

[34] Sasaki M, Hamada H, Kintaichi Y, Ito T. Application of a neural network to the analysis of catalytic reactions Analysis of NO decomposition over Cu/ZSM-5 zeolite. Appl Catal A, Gen. 1995 Nov 23;132(2):261-270.

[35] Cundari TR, Deng J, Zhao Y. Design of a propane ammoxidation catalyst using artificial neural networks and genetic algorithms. In: Industrial and Engineering Chemistry Research.

American Chemical Society; 2001. p. 5475-5480.

[36] Corma A, Serra JM, Argente E, Botti V, Valero S. Application of artificial neural networks to combinatorial catalysis: Modeling and predicting ODHE catalysts. ChemPhysChem. 2002;3(11):939-945.

[37] Valeh-E-Sheyda P, Yaripour F, Moradi G, Saber M. Application of artificial neural networks for estimation of the reaction rate in methanol dehydration. Ind Eng Chem Res. 2010 May 19;49(10):4620-4626.

[38] Günay ME, Yildirim R. Neural network Analysis of Selective CO Oxidation over Copper-Based Catalysts for Knowledge Extraction from Published Data in the Literature. Ind Eng Chem Res. 2011;50(22):12488-12500.

[39] Baumes L, Farrusseng D, Lengliz M, Mirodatos C. Using artificial neural networks to boost high-throughput discovery in heterogeneous catalysis. QSAR Comb Sci. 2004;23(9):767-778.

[40] Cavalcanti FM, Schmal M, Giudici R, Brito Alves RM. A Catalyst Selection Method for the Water-Gas Shift Reaction using Artificial Neural Networks. In: 1st Latin American Conference on Sustainable Development of Energy, Water and Environment Systems - LA SDEWES. Rio de Janeiro; 2018. p. 1-11.

[41] Garona HA, Cavalcanti FM, Abreu TF, Schmal M, Brito Alves RM. Using Artificial Neural Networks for Fischer-Tropsch Synthesis to Lower-Olefins Production Optimization. In: 15th Conference on Sustainable Development of Energy, Water and Environment Systems - SDEWES. Cologne; 2020. p. 1-15.

[42] Azzam M, Aramouni NAK, Ahmad MN, Awad M, Kwapinski W, Zeaiter J. Dynamic optimization of dry reformer under catalyst sintering using neural networks. Energy Convers Manag. 2018 Feb 1;157:146-156.

[43] Juybar M, Khanmohammadi Khorrami M, Bagheri Garmarudi A, Zandbaaf S. Determination of acidity in metal incorporated zeolites by infrared spectrometry using artificial neural network as chemometric approach. Spectrochim Acta - Part A Mol Biomol Spectrosc. 2020 Mar 5;228:117539.

[44] Schmal M. Heterogeneous Catalysis and its Industrial Applications. 1st ed. Switzerland: Springer; 2016.

[45] Assidjo E, Yao B, Kisselmina K, Amané D. Modeling of an industrial drying process by artificial neural networks. Brazilian J Chem Eng. 2008;25(3):515-522.

[46] Fernandes FAN, Lona LMF. Neural network applications in polymerization processes. Brazilian J Chem Eng. 2005;22(3):401-418.

[47] Alves RMB, Nascimento CAO. Gross errors detection of industrial data by neural network and cluster techniques. Brazilian J Chem Eng. 2002;19(4):483-489.

[48] Cai Q, Lee BCY, Ong SL, Hu J. Application of a Multiobjective Artificial Neural Network (ANN) in Industrial Reverse Osmosis Concentrate Treatment with a Fluidized Bed Fenton Process: Performance Prediction and Process Optimization. ACS ES&T Water. 2021;0-11.

[49] Lin M, Wu Y, Rohani S. Simultaneous Measurement of Solution Concentration and Slurry Density by Raman Spectroscopy with Artificial Neural Network. Cryst Growth Des [Internet]. 2020 Mar 4 [cited 2021 Feb 8];20(3):1752-9. Available from: https://dx.doi.org/10.1021/acs.cgd.9b01482

[50] Douglas JM. Conceptual Design of Chemical Processes. Vol. 1. New York: McGraw-Hill; 1988. 1110 p.

[51] Yeomans H, Grossmann IE. A systematic modeling framework of superstructure optimization in process synthesis. Comput Chem Eng. 1999;23(6):709-731.

[52] Graciano JEA, Le Roux GAC. Improvements in surrogate models for process synthesis. Application to water network system design. Comput Chem Eng. 2013;59:197-210.

[53] Mencarelli L, Chen Q, Pagot A, Grossmann IE. A review on superstructure optimization approaches in process system engineering. Comput Chem Eng. 2020;136:106808.

[54] Biegler LT, Grossmann IE, Westerberg AW. Systematic Methods of Chemical Process Design. Upper Saddle River, NJ: Prentice Hall PTR; 1999. 808 p.

[55] Grossmann IE, Daichendt MM. New trends in optimization-based approaches to process synthesis. Comput Chem Eng. 2003;20(6-7):665-683.

[56] Ryu J, Kong L, Pastore de Lima AE, Maravelias CT. A generalized superstructure-based framework for process synthesis. Comput Chem Eng. 2020;133:106653.

[57] Haykin S. Neural networks: a comprehensive foundation. Prentice Hall PTR; 1994.

[58] Henao CA, Maravelias CT. Surrogate-based process synthesis. Vol. 28, Computer Aided Chemical Engineering. Elsevier B.V.; 2010. 1129-1134 p.

[59] Savage T, Almeida-Trasvina HF, del Río-Chanona EA, Smith R, Zhang D. An adaptive data-driven modelling and optimization framework for complex chemical process design. Comput Aided Chem Eng. 2020;48:73-78.

[60] Klemeš J, Friedler F, Bulatov I, Varbanov P. Sustainability in the Process Industry: Integration and Optimization. New York, Chicago, San Francisco, Lisbon, London, Madrid, Mexico City, Milan, New Delhi, San Juan, Seoul, Singapore, Sydney, Toronto: McGRAW-HILL; 2011. 385 p.

[61] Nascimento CAO, Giudici R, Guardani R. Neural network based approach for optimization of industrial chemical processes. Comput Chem Eng. 2000 Oct 1;24(9-10):2303-2314.

[62] Cai QQ, Lee BCY, Ong SL, Hu JY. Fluidized-bed Fenton technologies for recalcitrant industrial wastewater treatment–Recent advances, challenges and perspective. Vol. 190, Water Research. Elsevier Ltd; 2021. p. 116692.

[63] Alsaffar MA, Ghany MARA, Ali JM, Ayodele BV, Mustapa SI. Artificial Neural Network Modeling of Thermo-catalytic Methane Decomposition for Hydrogen Production. Top Catal [Internet]. 2021 Jan 2 [cited 2021 Feb 8];1:3. Available from: https://doi.org/10.1007/s11244-020-01409-6

[64] Md Nor N, Che Hassan CR, Hussain MA. A review of data-driven fault detection and diagnosis methods: Applications in chemical process systems. Rev Chem Eng. 2020;36(4):513-553.

[65] Luo L, Xie L, Su H. Deep Learning with Tensor Factorization Layers for Sequential Fault Diagnosis and Industrial Process Monitoring. IEEE Access. 2020;8:105494-105506.

[66] Gao X, Yang F, Feng E. A process fault diagnosis method using multi-time scale dynamic feature extraction based on convolutional neural network. Can J Chem Eng. 2020;98(6):1280-1292.

[67] Peng C, Lu RW, Kang O, Kai W. Batch process fault detection for multi-stage broad learning system. Neural Networks. 2020;129:298-312.

[68] Kamat S, Madhavan KP. Developing ANN based virtual/soft sensors for industrial problems. IFAC-PapersOnLine. 2016;49(1):100-105.

[69] Zhang Z, Wu Z, Rincon D, Christofides PD. Real-time optimization and control of nonlinear processes using machine learning. Mathematics. 2019;7(10):1-25.

[70] Wu Z, Tran A, Ren YM, Barnes CS, Chen S, Christofides PD. Model predictive control of phthalic anhydride synthesis in a fixed-bed catalytic reactor via machine learning modeling. Chem Eng Res Des. 2019;145:173-183.

[71] Chen S, Wu Z, Christofides PD. A cyber-secure control-detector architecture for nonlinear processes. AIChE J. 2020;66(5):1-18.

[72] Hernavs J, Ficko M, Berus L, Rudolf R, Klančnik S. Deep Learning in Industry 4 . 0 – Brief Overview. J Prod Eng. 2018;21(2):1-5.

[73] Duever TA. Data science in the chemical engineering curriculum. Processes. 2019;7(11).

Chapter 10

Material Classification via Machine Learning Techniques: Construction Projects Progress Monitoring

Wesam Salah Alaloul and Abdul Hannan Qureshi

Abstract

Nowadays, the construction industry is on a fast track to adopting digital processes under the Industrial Revolution (IR) 4.0. The desire to automate maximum construction processes with less human interference has led the industry and research community to inclined towards artificial intelligence. This chapter has been themed on automated construction monitoring practices by adopting material classification via machine learning (ML) techniques. The study has been conducted by following the structure review approach to gain an understanding of the applications of ML techniques for construction progress assessment. Data were collected from the Web of Science (WoS) and Scopus databases, concluding 14 relevant studies. The literature review depicted the support vector machine (SVM) and artificial neural network (ANN) techniques as more effective than other ML techniques for material classification. The last section of this chapter includes a python-based ANN model for material classification. This ANN model has been tested for construction items (brick, wood, concrete block, and asphalt) for training and prediction. Moreover, the predictive ANN model results have been shared for the readers, along with the resources and open-source web links.

Keywords: automated progress tracking, artificial intelligence, ANN, construction sector

1. Introduction

The construction progress measuring practices are considered indispensable tools for effective project control [1]. Efficient and effective progress monitoring practices provide information regarding performance deviations to the execution plan and help the project management office (PMO) towards timely implementation of control actions to minimise the negative impacts [2]. Currently, instead of manual practices for construction progress assessment, the research community is fascinated by techniques such as photogrammetry, laser scanning, time-lapse photography, etc. Moreover, these strategies have also adopted 4D Building Information Modelling (BIM) as a framework to execute model-based progress tracking of construction projects [3]. In the last two decades, advancements in computer processes and digital camera technologies have allowed construction sector to effectively process [4] and retrieve valuable information from video clips and digital images. Moreover, the applications of computer vision (CV) and image processing systems are now considered in the architecture, engineering and construction (AEC) industry as an emerging field of research with steady growth [5]. Whereas, automated building material classification has gained the interest of the research community with respect to the AEC industry. Automated material classification may increase the performance output of various activities, including defect identification, on-site material control and progress monitoring [6]. The as-is BIM includes the geometric as well as non-geometric information on the building components, including the building materials, which is necessary for energy simulations and 3D structure visualisations. This evolution has led machine learning (ML) techniques to gain popularity for material classification models. Material classification can be performed via laser scan data and image-based data detection; however, the latter is more popular among the research community. The general concept of image-based approaches relies on utilising visual characteristics of building materials such as projection, roughness, colour and shape for automated detection. However, image-based approaches are highly affected by lighting environment. The varying light conditions have a substantial impact on the visual properties of the materials, which create difficulties in classifying the image-based building material. Moreover, the weak textures on surfaces and uncertain points of view often adversely affect the effectiveness and precision process of image-based content classification [7, 8]. Material classification is considered a vital activity of any vision-based framework to generate conceptual as-built 3D models for automatic progress monitoring in construction projects. In the case of construction material, related details can be obtained primarily from the appearance-based details found in 2D images. Digitalised material classification extracts the appearance-based information for construction progress tracking and perform segmentation process for the effective generation of automated 3D as-built models [9].

The implementation of ML techniques is vital for dynamic operations of the systems with continuous and automated learning [10]. Other than pattern recognition, ML technologies are adopted for the self-learning of the big-data based systems connected via the internet of things (IoT) integrated with digital technologies [11]. Likewise, for the construction progress detection technologies, the trend of integration with ML techniques for the digitalisation of the monitoring process has also been increased in recent times [12]. ML algorithms are generally divided into supervised and unsupervised types, which are analysed for learning and prediction of empiric results. Supervised learning algorithms minimise the error between the targeted data and output data, whereas unsupervised algorithms are adopted for clustering data when data training is not preferable. However, both types of ML algorithms can be utilised for the material classifications based on the site conditions and circumstances for the availability of input data [13]. Construction material classification via ML techniques has gained a lot of attention among professionals and researchers in the construction sector. Various studies can be found related to material classification for construction progress monitoring. However, still, improvements are required in the methodologies and algorithms towards effective and efficient outcomes. Therefore, this chapter aims to overview the applications of construction material classifications via ML techniques for progress monitoring/tracking/detection of construction projects by conducting a short systematic review. Moreover, a simple artificial neural network (ANN) based material classification algorithm has also been discussed at the end of this chapter, which will help the readers to understand practical implications of ML techniques in the construction sector.

2. Research methodology

For the achievement of the study objective, i.e., overview the applications of material classification via ML techniques in the construction progress monitoring domain, the literature was collected from Web of Science (WoS) and Scopus for the last ten years. Two different keywords combinations were designed for the collection of studies from the aforementioned databases. The first keywords combination was designed to explore overall automated construction project monitoring technologies, and the final results were sorted for material classification techniques via ML. The second keywords combination was designed specifically for automated construction monitoring practices using material classification via ML techniques. The study's scope was narrowed down to journal articles and building construction projects, for the last ten years data, i.e. 2010 to 2020. **Figure 1** shows the study flowchart of the adopted methodology for this chapter.

Using the first keywords combination, overall 54 studies were collected on construction automated progress monitoring technologies, out of which four studies were based on material classification via ML techniques. The summary of the data searching and collection for the first keywords combination is shown in **Table 1**.

Likewise, with the second keywords combination, 48 studies were collected and out of which ten were found relevant to the designed scope of this chapter. **Table 2** shows the summary of the data searching and collection for the second keywords combination.

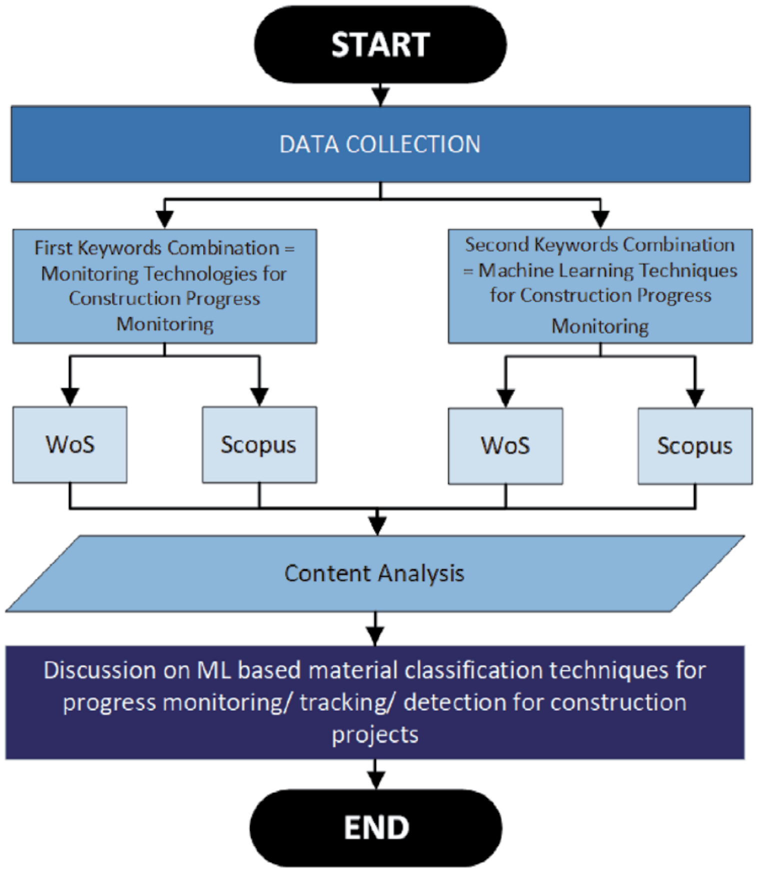

START
DATA COLLECTION
First Keywords Combination = Monitoring Technologies for Construction Progress Monitoring
Second Keywords Combination = Machine Learning Techniques for Construction Progress Monitoring
WoS
Scopus
WoS
Scopus
Content Analysis
Discussion on ML based material classification techniques for progress monitoring/ tracking/ detection for construction projects
END

Figure 1.
Study methodology framework.

Database	Duration	Keywords combination	Total collected papers	Relevant papers
WoS	2010–2020	"TS = (automat* AND (construction OR project OR progress) AND (monitor* OR updat* OR track* OR detect* OR recogn*))"	54	4
Scopus	2010–2020	"TITLE-ABS-KEY (automat* AND (construction OR project OR progress) AND (monitor* OR updat* OR track* OR detect* OR recogn*))"		

Table 1.
Data collection summary of material classification-ML techniques with first keywords combination.

Database	Duration	Keywords combination	Total collected papers	Relevant papers
WoS	2010–2020	"TS = (automat* AND (construction OR project OR progress) AND (monitor* OR updat* OR track* OR detect* OR recogn* OR classification) AND (machine learning OR ML OR ANN OR artificial neural network))"	48	10
Scopus	2010–2020	"TITLE-ABS-KEY (automat* AND (construction OR project OR progress) AND (monitor* OR updat* OR track* OR detect* OR recogn* OR classification) AND ("machine learning" OR ml OR ann OR "artificial neural network"))"		

Table 2.
Data collection summary of material classification-ML techniques with second keywords combination.

Overall, 14 studies were found relevant to the defined scope, which were further analysed for the in-depth review.

3. Discussion

Material classifications via ML techniques are popular among the research community in every domain. However, in the construction progress monitoring practices, the material classification via ML is still an emerging area. Collection of data from WoS and Scopus supports this argument as with the first keywords combination out of 54 technical journal based studies, only four were related to material classification for construction progress monitoring. Moreover, with the second keywords combination out of 48 ML studies in the construction progress monitoring domain, only ten were related to material classification. There are various ML classifiers which are being adopted by the researchers such as random forest (RF), decision tree (DT), bayesian, k-nearest neighbours (KNN), gaussian mixture modes (GMM), logistic regression (LR), support vector machine (SVM), and artificial neural networks (ANN), etc. However, ANN and SVM are the most favourite techniques among researchers, when it comes to material classification [5, 8, 14]. The material classification has been performed by researchers on various sources data input such as digital images taken with the help of a camera [15], smartphones, drones [16], and 3D point cloud models generated on collected images via structure from motion (SfM) [17], or laser scanners [8].

Table 3 illustrates a general summary of the collected studies for construction progress monitoring by adopting material classification via ML techniques.

Ref	Year	Data input	Adopted techniques	Materials classified
[18]	2010	Site Images	ANN, SVDD, & C-SVC	Concrete
[19]	2010	Spectral Information	FSVM with a PUK kernel	Granite
[20]	2012	Site Images	GMM, ANN, & SVM	Concrete
[14]	2014	Concrete Images	ANN, SVM, KNN, Bayesian, & FLD	Concrete
[9]	2014	Material Images	C-SVM (SVM)	Grass, Form Work, Marble, Gravel, Foliage, Soil-Loose, Soil-Compact, Paving, Soil-Vegetation, Stone-Granular, Soil-Mulch, Wood, Stone-Limestone, Brick, Cement-Smooth, Cement-Granular, Asphalt, Concrete-Precast, Concrete-Cast, & Metal-Grills
[3]	2015	Material Images	C-SVM (SVM)	Brick, Asphalt, Concrete, Foliage, Granular & Smooth Cement based surfaces, Gravel, Formwork, Marble, Insulation, Paving, Metal, Soil, Waterproofing Paint, Stone, & Wood
[5]	2016	Site Images	MLP, ANN, & RBF	Concrete, OSB Boards, & Red Brick
[17]	2016	3D Point Cloud/ Material Images	SVM	Windows, Walls, Protrusions, Tile, Brick, Stone, & Coating
[16]	2017	Site Images	CDF, SVM, RBF, LBP, & PPHT	Insulation, Studs, Outlets, Electrical & Three States for Drywall Sheets (Installed, Painted, & Plastered)
[21]	2019	Site Images	CNN (ANN)	Structural Elements
[22]	2019	Rebar Images	RFSP, OTSUHT, & IVB	Rebar
[23]	2020	Material Images	ANN	Sandstorms, Paving, Gravel, Stone, Cement-Granular, Brick, Soil, Wood, Asphalt, Clay Hollow Block, & Concrete Block
[8]	2020	Laser Scan Data/Material Images	DT, DA, NB, SVM, & KNN	Mortar, Concrete, Metal, Stone, Wood, Painting, Plastic, Plaster, Ceramic, & Pottery
[13]	2020	Proprioceptive Force Data	ANN, KNN, & k-mean	Rock and gravel

Table 3.
Summary of material classification studies for construction progress monitoring of building projects.

The in-depth review was performed on the collected studies, and it can be observed that material classification methodologies are being adopted for construction progress monitoring practices since before 2010. Zhu and Brilakis [18] identified concrete material regions by testing three classifiers, i.e., C-support vector classification (C-SVC), support vector data description (SVDD), and ANN, where ANN model was found with better outcomes. The model was developed using C++ and evaluated more than hundreds of building construction site digital images. Araújo et al. [19] adopted a functional SVM (FVSM) with a pearson VII function (PUK) kernel and linear functional regression for classifying granite varieties by using spectrophotometer spectrum data. Son et al. [20] developed a model for the identification of concrete structural elements in the coloured images. In the process, the red-green-blue (RGB) colour space was transformed to non-RGB colour spaces to enhance the separability between background classes and concrete. The model was tested for three ML algorithm, i.e., GMM, ANN, and SVM. However, SVM along with hue-saturation-and-intensity (HSI) colour space was found more effective. Yazdi and Sarafrazi [14] used five different classifiers, ANN, SVM, bayesian, KNN, and fisher's linear discriminate (FLD) algorithm in combinations and as separate, i.e. Bayesian, Bayesian + FLD, KNN, KNN + FLD, SVM, SVM + FLD, ANN, and ANN + FLD on the segmented concrete images. This study found the ANN model as the better option for automatic image segmentation. Dimitrov and Golparvar-Fard [9] proposed C-support vector machine (C-SVM) algorithm combined with texture and hue-saturation-value (HSV) colour features. This study tested the developed algorithm on 20 construction materials (paving, grass, gravel, stone-limestone, formwork, soil-vegetation, marble, metal-grills, soil-mulch, soil-compact, stone-granular, wood, soil-loose, asphalt, cement-granular, brick, concrete-cast, cement-smooth, foliage, and concrete-precast) with more than 150 images for each category and compared various pixel sizes (n x n), i.e., 30, 50, 75, 100, 150, and 200. Better accuracy and effective output were reported for 200 x 200 pixel-images. Han and Golparvar-Fard [3] developed a construction material library (CML) based on C-SVM classifiers with linear x^2 kernels on 100×100, 75×75, and 50×50 pixel-images datasets of cement-based surfaces, paving, brick, asphalt, formwork, foliage, concrete, marble, gravel, insulation, metal, soil, wood, stone, and waterproofing paint. This study developed an appearance-based material classification technique for progress monitoring using daily photologs and BIM. Point cloud models were generated using SfM and multi-view-stereo (MVS) algorithms from the construction site images. These point cloud models were superimposed with 4D BIM models, and registered site images were back-projected for the BIM elements for extracting related image patches. Testing of the extracted patches was performed with multi-class material classification technique that was pre-trained with the extended CML dataset. Rashidi et al. [5] conducted a comparison study on SVM, radial basis function (RBF), and multilayer perceptron (MLP) by evaluating the performance on building construction materials, i.e. OSB boards, red brick, and concrete. The feature extraction was performed from image blocks to compare the efficiency for detecting building construction materials, and SVM classifier with RBF kernel results were found more precise in perceiving the images for material textures. Yang et al. [17] performed material recognition of windows, walls, protrusions, tile, brick, stone, and coating using image-based 3D modelling. SfM was used for generating the 3D point cloud model for site images. The building facade was modelled as combined planes of protrusions, windows, and wall. Planes were primarily detected from 3D point clouds using random sample consensus (RANSAC) and further recognised as distinct structural components by SVM classifier. Hamledari et al. [16] adopted CV integrated with shape and colour-based modules that automatically detect the interior components using 2D digital images, i.e. three states drywall sheets (plastered, installed, and painted), insulation, studs, and electrical outlets. Cumulative distribution function (CDF) used for electrical outlet module, SVM classifier has an RBF kernel type for drywall, local binary patterns (LBP) for insulation module, and progressive probabilistic hough transform (PPHT) for stud module. The method was validated by indoor construction site images captured by unmanned aerial vehicle (UAV), smartphone and internet sources. Braun and Borrmann [21] developed an automatically labelling process via construction images with 4D BIM and 3D point cloud approach. The 3D point cloud model was integrated with the BIM model, and automated labelling for structural elements was provided with the semantic information. The convolutional neural network (CNN) model was trained on this information to generate classification tasks.

The accuracy of the allocated labels was checked by pixel-based field comparison to manual labels. Lee and Park [22] developed automatic reinforcing-bar image analysis system (ARIAS), which could separate the background for the bar area calculation. The model was also able to count the number of bars by testing various combinations between RF and super-pixel method (RFSP), otsu threshold for extracting the bars areas (OTSU), hough transforms (HT), and iterative voting for binary image (IVB). The combination RFSP+IVB gave better output results than other combinations. Ghassemi et al. [23] proposed the material classification model based on deep learning (DL) algorithm for classifying in various illumination conditions and different camera angles/positions. Eleven construction materials (sandstorms, paving, gravel, stone, cement-granular, brick, soil, wood, asphalt, clay hollow block, and concrete block) were classified in this study, and good accuracy was achieved by VGG16 algorithm, even for images that were hard to identify by the human eye. Yuan et al. [8] proposed an automatic material classification method with the help of 2D digital images using the graphical characteristics of building materials. A coloured laser scan data was generated using a terrestrial laser scanner (TLS) with a built-in camera, which contained the surface geometries, material reflectance and surface roughness of building materials. TLS data were classified using DT, Discriminant Analysis (DA), Naive Bayes (NB), SVM, and KNN. A laser scan database for ten common construction building materials (stone, pottery, mortar, concrete, ceramic, wood, plastic, plaster, metal, and painting) was used to train and validate the model. However, better results were achieved from one-class SVM (OC-SVM) and SVDD. Fernando and Marshall [13] performed a state of the art classification methodology for rock and gravel, by identifying force data (proprioceptive force data) acquired from load-haul-dump equipment with capacity of 14-tonne and adopting ANN, KNN, and k-mean algorithms. However, good results were obtained by ANN (5-NN) model with more realistic classification. Since the system only relies on proprioceptive sensing, it is feasible in harsh, dusty, and dark conditions that hinder the use of external sensor data. **Table 4** exemplifies the

Ref	Adopted technique	Main features	Achieved outcomes	Remarks/observed limitations
[18]	ANN, SVDD, & C-SVC	The model was trained by datasets of negative & positive concrete images.	ANN technique was found better with an average precision of 83.3%, average recall of 79.6%, & overall concrete detection of 80%.	The accuracy of the model outcome was evaluated by manually tracing the concrete image pixels.
[19]	Functional linear regression, & FSVM with a PUK kernel	Granite varieties were characterised using a spectrophotometer (vectorial spectral information).	FSVM with a PUK kernel was found better with 0.82% validation error rate.	For assessing the total real colour of the stone, the data was to be collected from various points on the sample.
[20]	GMM, ANN, & SVM	108 images were collected for 50 construction projects on different timings & weather conditions.	The SVM model with the HSI colour space was found better with an accuracy rate of 91.68%.	The detection performance was evaluated by identifying concrete & background pixels.
[14]	ANN, SVM, KNN, Bayesian, & FLD	A dataset contained 31 images of concrete with the image resolution of 2 mm.	ANN was the better choice for automatic image segmentation with correctly classified pixels up to 90.29%.	This study covered the segmentation of concrete images.

Ref	Adopted technique	Main features	Achieved outcomes	Remarks/observed limitations
[9]	C-SVM (SVM)	Various pixels sizes were tested, i.e., 30 × 30, 50 × 50, & 200 × 200. Material datasets were created for varying degrees of viewpoint, illumination, and scales.	97.1% average classification rate was achieved for 200 × 200 pixel images.	This study did not cover the challenge of segmentation.
[3]	C-SVM (SVM)	This study adopted BIM integrated daily construction photologs for extraction relevant image patches. Study adopted materials library for images with three different patch sizes of 50 × 50, 75 × 75, & 100 × 100.	The accuracy of 92.4% was achieved for the dataset with 100 × 100 pixel size images.	Practical limitations for adopted methodology include comprehensiveness of materials library, completeness of 3D reconstruction, and computation time.
[5]	MLP, ANN, & RBF	For feature extraction, the construction materials were divided into three groups: 1) materials encompasses a very distinct colour, 2) variable colour patterns, & 3) material without a distinctive colour pattern.	SVM classifier with RBF kernel was found better than other techniques.	Outcomes may be affected due to lack of adequate light, varying viewpoint angle, & image capturing distance.
[17]	SVM	The study performed the material detection method & image-based 3D modelling. The 3D model was generated via SfM and segmented into planar components, which were recognised as structural components via knowledge-based reasoning.	RANSAC was adopted for detection in the point cloud. The model achieved an average accuracy of 95.55%.	The datasets consist of 463 stone samples, 637 brick samples, 504 tile samples, & 409 coating samples.
[16]	CDF, SVM, RBF, LBP, & PPHT	Input source data was tested for UAV, smartphones & internet sources. A CV technique was adopted for the detection of interior components & extrapolated the existing scenario via 2D digital images.	Three datasets were adopted, and the following outcomes were attained: 1) for stud module, precision above 90%, & recall above 83%. 2) for insulation module, precision above 88%, & recall above 89%. 3) for electric outlet module, precision above 86%, & recall above 87%.	Practical limitations for adopted methodology include the inability to detect & metallic electrical boxes, partitions with low visibility, & improperly captured scenes.
[21]	CNN (ANN)	The study follows an automated construction materials' labelling process of construction site images. The model combines the available information from the photogrammetric model & the 4D BIM. By aligning the BIM & 3D point cloud, a digital components can be projected onto the image.	91% pixel-wise accuracy was validated by the sample.	The construction & model inaccuracies, errors in post estimation during SfM, large scale deviations for real world coordinates, & occlusions may cause labelling errors.

Ref	Adopted technique	Main features	Achieved outcomes	Remarks/observed limitations
[22]	RFSP, OTSUHT, & IVB	The model performs analysis on the reinforcing bars of the production plant moving along a conveyor belt by accurately calculating the bar area and its number.	RFSP+IVB gave better results with 0.89 as F-score.	
[23]	ANN	The method uses the DL model by data augmentation & prevents over-fitting of network structures for images with varying camera resolution, illumination, & small datasets.	VGG16 algorithm gave the maximum accuracy of 97.35%.	Raspberry Pi 3 was used with datasets taken from different construction sites with 1231 images of 11 classes for various views of materials.
[8]	DT, DA, NB, SVM, & KNN	The TLS based laser scan data can provide information for building material for surface geometries such as surface roughness and material reflectance.	OC-SVM and SVDD gave better results with an average classification accuracy of 96.7%.	In this study, only plane target surfaces were considered.
[13]	ANN, KNN, & k-mean	The study followed a material classification methodology using proprioceptive force data acquired from an digging machine integrated with ML. The model is pertinent in the dusty, dark and harsh areas.	ANN model was found effective with a classification accuracy of 90%.	The model only covers rock & gravel, where excavated soil may consist of various other materials.

Table 4.
Technical summary of collected articles for progress monitoring via ML of the building construction projects.

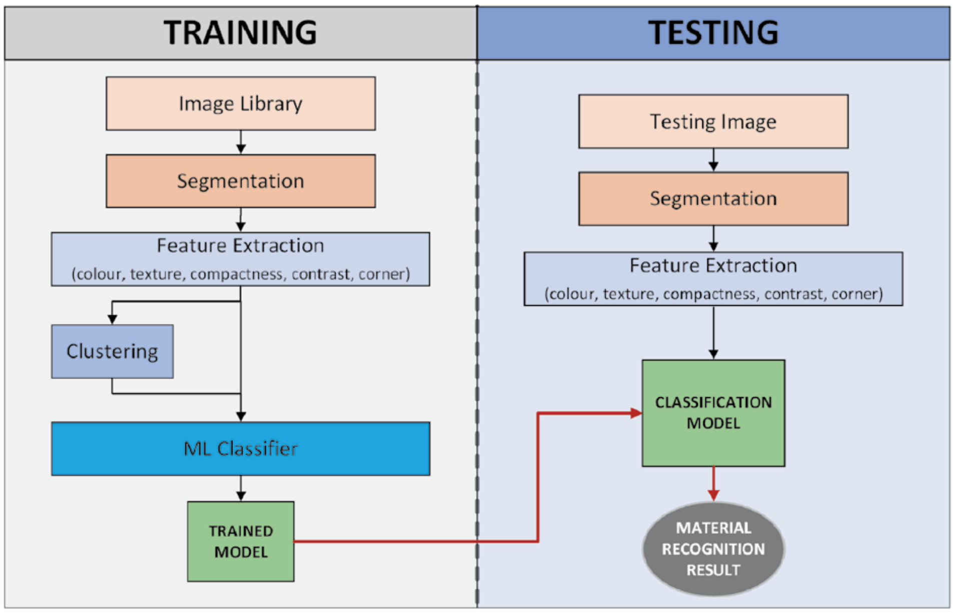


Figure 2.
General workflow of ML-material classification models.

technical summary of collected studies for their main features, achieved outcomes, and observed limitation if any.

The general workflow of ML material classification model, via images, is shown in **Figure 2**. In the training process, the model is usually trained with the help of datasets of construction material images. The segmentation process is performed on the collected images, as segmentation increases the performance output of the model as compared to non-segmented images [24]. GMM technique can also be utilised for image segmentation. The feature extraction is performed on the segmented images for their colour, texture, compactness, contrast. Researchers mostly have adopted HSV, RGB (red, green and blue), Image patch (IP), 48-dimensional form (LM), and 18-dimensional rotationally invariant form (rLM) for features extraction and found better results with LM + HSV combination [9]. Depending upon the type of model, clustering process can be adopted, and on this data any ML classifier can be applied for model training. While testing the model for any random material image, the segmentation and feature extraction processes are performed, which are further identified with the help trained ML model for final material recognition output.

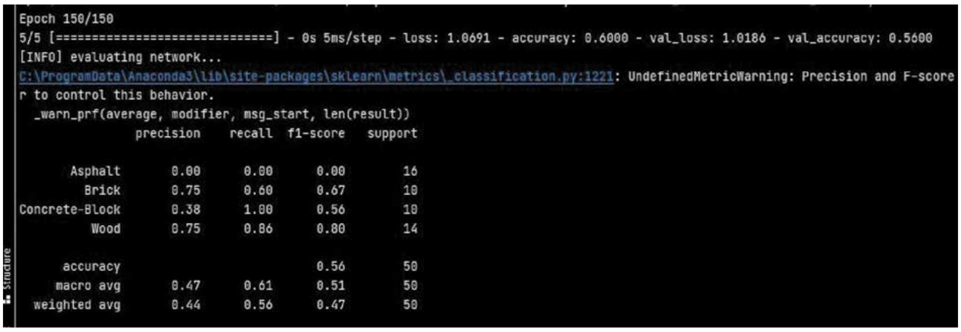
4. Python-based ML-material classification model

In this section of the chapter, a small exercise has been performed to detect construction materials via ML algorithm for the preview of the readers to the practical application of ML in material classifications for construction progress monitoring. The exercise has been executed by using the resources available as open-source on the internet. The simple ANN model has been adopted, developed by Adrian Rosebrock [25] using Python under Keras and TensorFlow environment [26]. Moreover, the construction materials datasets for the images have been collected from the open-source GitHub repository for concrete blocks, asphalt, wood, and bricks [27]. The selected model performed the classification of construction materials in two phases. In the first phase, the model has been trained with the help of collected datasets, and in the same phase, validation of the model has been performed. The adopted ANN-based model splits 75% data for training purpose and the remaining 25% data for validation/testing. In the training phase of the model, the images of each construction item have been placed in the separate folders labelled as the name or ID of that particular construction material. The model analyses the colour, texture and geometrical aspects of images under each construction material ID and trains its memory for each construction material separately against the given name or ID. The same model validates or tests itself on the assigned images, to verify the model effectiveness for the predictions. If the model fails to give an effective validation run, either model needs to be trained more or model structure needs to be reviewed. In the second phase, the model has been applied to predict and identify construction materials on randomly selected images from the internet. The predictive model uses the memory of the trained model to predict construction item on the input image. Moreover, for improving the training model, the number of ANN hidden layers and epochs per run can be increased in the model, which enhances the output results for accuracy and decreases the data loss [28]. Therefore, to observe the effects of varying epochs per run on the predictive model, the performance of the model has been tested on two different scenarios of epochs per run, i.e., 150 and 300.

4.1 Model training and testing

The training of the ANN model was performed on the images datasets of each construction material, i.e., concrete blocks, asphalt, wood, and bricks, with each dataset comprised of 50 images. As the model has been designed to train on 75% of the provided data; therefore, for each construction item, the model was trained on 38 images. The remaining 12 images were used by the model for validation or testing. Two different models were trained on varying epochs per run, i.e., 150 and 300. The first model, with 150 epochs per run, attained the accuracy of 56% as shown in **Figure 3**, where its graphical representation for the attainment of model accuracy and loss can be seen in **Figure 4**. It can be seen that 'Asphalt' in this model got '0' (zero) value for precision and F1 score. The maximum precision and F1 score have been attained for wood (precision = 0.75, F1 score = 0.80). Whereas, the lowest precision and F1 score have been achieved by concrete block (precision = 0.35, F1 score = 0.56).

In the second model, the epochs per run for the model was set to 300, and 64% model accuracy was attained. The attained model accuracy, data loss along with precision, recall and F1 score can be seen in **Figure 5**, where the graphical representation of the training data for the attainment of model accuracy and loss for 300 epochs can be seen in **Figure 6**. The accuracy and F1 score of the second model,



```
Epoch 150/150
5/5 [==============================] - 0s 5ms/step - loss: 1.0691 - accuracy: 0.6000 - val_loss: 1.0186 - val_accuracy: 0.5600
[INFO] evaluating network...
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Precision and F-score
r to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
                precision    recall  f1-score   support

       Asphalt       0.00      0.00      0.00        16
         Brick       0.75      0.60      0.67        10
Concrete-Block       0.38      1.00      0.56        10
          Wood       0.75      0.86      0.80        14

      accuracy                           0.56        50
     macro avg       0.47      0.61      0.51        50
  weighted avg       0.44      0.56      0.47        50
```

Figure 3.
Training/testing model output for 150 epochs per run.

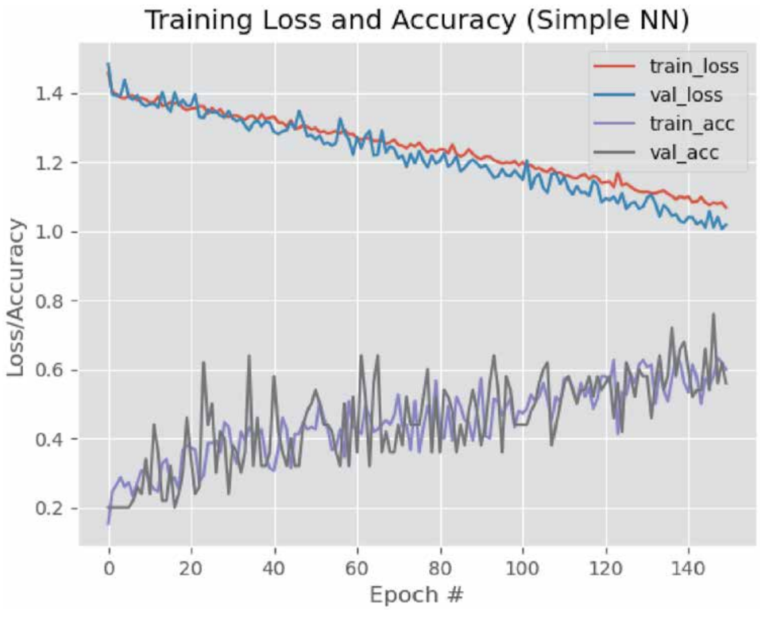
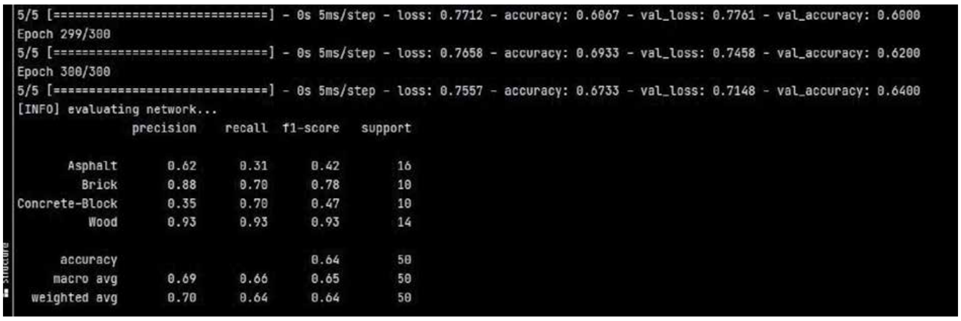


Figure 4.
Graphical representation of model accuracy and loss for 150 epochs per run.



```
5/5 [==============================] - 0s 5ms/step - loss: 0.7712 - accuracy: 0.6067 - val_loss: 0.7761 - val_accuracy: 0.6000
Epoch 299/300
5/5 [==============================] - 0s 5ms/step - loss: 0.7658 - accuracy: 0.6933 - val_loss: 0.7458 - val_accuracy: 0.6200
Epoch 300/300
5/5 [==============================] - 0s 5ms/step - loss: 0.7557 - accuracy: 0.6733 - val_loss: 0.7148 - val_accuracy: 0.6400
[INFO] evaluating network...
                precision    recall  f1-score   support

       Asphalt       0.62      0.31      0.42        16
         Brick       0.88      0.70      0.78        10
Concrete-Block       0.35      0.70      0.47        10
          Wood       0.93      0.93      0.93        14

      accuracy                           0.64        50
     macro avg       0.69      0.66      0.65        50
  weighted avg       0.70      0.64      0.64        50
```

Figure 5.
Training/testing model output for 300 epochs per run.

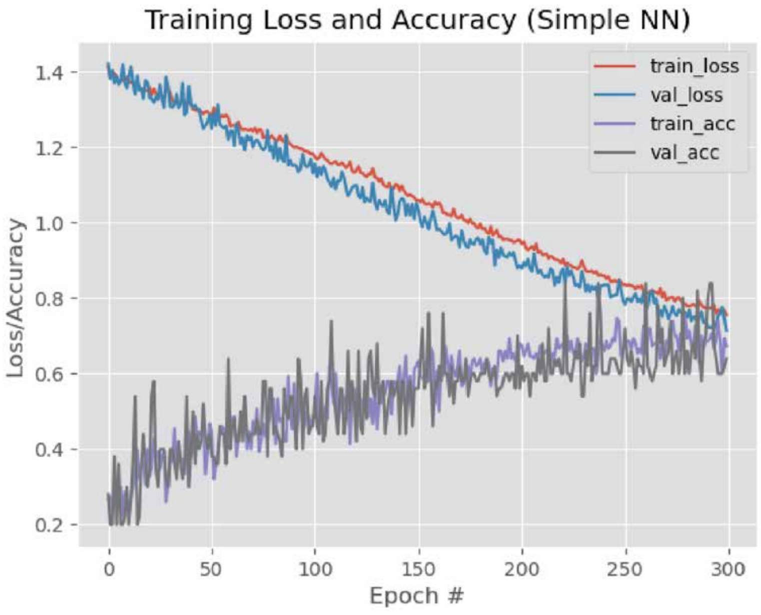


Figure 6.
Graphical representation of model accuracy and loss for 300 epochs per run.

with the increased epochs, was better than the first model, which validates the enhanced performance of the ANN on increasing the epochs per run.

It can be seen that maximum precision and F1 score have been attained for wood (precision = 0.93, F1 score = 0.93), and brick (precision = 0.88, F1 score = 0.78. Whereas, the lowest precision has been achieved by concrete block (precision = 0.35) and F1 score by asphalt (F1 score = 0.42).

4.2 Predictive model

The output of the predictive model is much dependent on the training of the model. For the testing of the predictive model, random images were collected from the internet for each. i.e., concrete block, asphalt, wood and brick. The selected predictive model utilised the memory output of the trained model and predicted the material along with its probability. The model predictions were observed for both trained models, i.e., 150 epochs and 300 epochs. **Table 5** shows the summary and comparison of the predictive model against the input images for both trained

Construction material	Input image	Model prediction for 150 epochs	Model prediction for 300 epochs
Brick		Wood probability = 36.13% (inaccurate)	Brick probability = 42.40% (accurate)
Wood		Wood probability = 63.63% (accurate)	Wood probability = 93.08% (accurate)
Concrete Block		Concrete Block probability = 36.20% (accurate)	Concrete Block probability = 43.70% (accurate)
Asphalt		Wood probability = 47.52% (inaccurate)	Asphalt probability = 31.08% (accurate)

Table 5.
Summary of predictive model.

models (150 epochs and 300 epochs) along with the model prediction probability percentage.

It can be illustrated from the results that ANN model performance has been enhanced and increased by increasing epochs per runs. Model predictions for the construction materials against the training model with 150 epochs were mostly inaccurate, which can be seen in 'Brick' and 'Asphalt'. Whereas, for the same input images, the model predictions were accurate and reliable with more prediction probability against the trained model with 300 epochs. Thus, the outcome of the predictive model is dependent on the structure of the trained ANN model.

5. Conclusions

The emergence of automation and IoT, in the construction sector, have regained the interest of professional and research community towards ML techniques. The ML technologies are now being adopted in many construction processes and one of which is construction progress monitoring. The theme of this chapter was designed to overview the application of material classification via ML techniques and their implication in the construction automated progress monitoring. For the achievement of this study objective, a small structured review was performed to collect relevant studies from WoS and Scopus. Overall, 14 studies were found relevant, where the majority of studies were performed for the multi-classification of construction materials using digital images. Moreover, the classification of material has also been performed based on proprioceptive force data. ANN and SVM models have been found most effective ML techniques for classification, and these techniques have also been integrated with BIM for effective construction processes control. For the better understanding of the readers to the practical implementation of ML techniques, a small experiment for multi-classification of construction materials (brick, asphalt, concrete block, and wood) using Python has also been included in this chapter. A simple ANN-based model was trained on the dataset of the aforementioned construction materials, and predictions were performed on random images. The utilised resources were collected from the open-source repositories, and details of their parent websites have also been shared for the readers' interests.

In this era of automation, the construction sector is inclined towards the adoption of artificial intelligence, and ML is playing a vital role in the enhancement of construction processes in various ways. Likewise, material classification is one of the favourite techniques when it comes to ML, especially for project progress monitoring. Although various studies have been conducted, however, still, there is a need for dedicated research to improve algorithms and methodologies to make these construction monitoring processes more effective and feasible for construction stakeholders.

Notes

The resources and program codes used in this chapter for the Python-based material classification model are available as open-source, and their access links are provided for the readers.

Abbreviations

IoT	Internet of Things
ML	Machine learning
ANN	Artificial neural network
PMO	Project management office
BIM	Building Information Modelling
AEC	Architecture, engineering, and construction
DT	Decision tree
RF	Random forest
LR	Logistic regression
KNN	K-nearest neighbours
GMM	Gaussian mixture modes
SVM	Support vector machine
SfM	Structure from motion
SVDD	Support vector data description
C-SVC	C-support vector classification
PUK	Pearson VII function
RGB	Red, green, blue
HSI	Hue, saturation, and intensity
CML	Construction material library
RANSAC	Random sample consensus
UAV	Unmanned aerial vehicle
CNN	Convolutional neural network
ARIAS	Automatic reinforcing-bar image analysis system
RFSP	Random forest and super-pixel method
IVB	Iterative voting for binary image
TLS	Terrestrial laser scanner
LM	48-dimensional form
rLM	18-dimensional rotationally invariant form

Author details

Wesam Salah Alaloul*† and Abdul Hannan Qureshi†
Department of Civil and Environmental Engineering, Universiti Teknologi PETRONAS, Tronoh, Perak, Malaysia

*Address all correspondence to: wesam.alaloul@utp.edu.my

† These authors contributed equally.

References

[1] Pazhoohesh M, Zhang C. Automated construction progress monitoring using thermal images and wireless sensor networks. GEN. 2015;101:01.

[2] Lin Z, Petzold F, Ma Z. A Real-Time 4D Augmented Reality System for Modular Construction Progress Monitoring. In: ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction. vol. 36. IAARC Publications; 2019. p. 743–748.

[3] Han KK, Golparvar-Fard M. Appearance-based material classification for monitoring of operation-level construction progress using 4D BIM and site photologs. Automation in Construction. 2015;53: 44–57. Available from: http://dx.doi.org/10.1016/j.autcon.2015.02.007.

[4] Li Y, Liu C. Applications of multirotor drone technologies in construction management. International Journal of Construction Management. 2019;19(5):401–412.

[5] Rashidi A, Sigari MH, Maghiar M, Citrin D. An analogy between various machine-learning techniques for detecting construction materials in digital images. KSCE Journal of Civil Engineering. 2016;20(4):1178–1188.

[6] Meroño JE, Perea AJ, Aguilera MJ, Laguna AM. Recognition of materials and damage on historical buildings using digital image classification. South African Journal of Science. 2015.

[7] Lu Q, Lee S, Chen L. Image-driven fuzzy-based system to construct as-is IFC BIM objects. Automation in Construction. 2018.

[8] Yuan L, Guo J, Wang Q. Automatic classification of common building materials from 3D terrestrial laser scan data. Automation in Construction. 2020; 110(October 2019):103017. Available from: https://doi.org/10.1016/j.autcon.2019.103017.

[9] Dimitrov A, Golparvar-Fard M. Vision-based material recognition for automated monitoring of construction progress and generating building information modeling from unordered site image collections. Advanced Engineering Informatics. 2014;28(1):37–49.

[10] Hadidi R, Cao J, Woodward M, Ryoo MS, Kim H. Real-time image recognition using collaborative iot devices. In: Proceedings of the 1st on Reproducible Quality-Efficient Systems Tournament on Co-designing Pareto-efficient Deep Learning; 2018. p. 1.

[11] Jameel SM, Hashmani MA, Rehman M, Budiman A. An adaptive deep learning framework for dynamic image classification in the internet of things environment. Sensors (Switzerland). 2020;20(20):1–25.

[12] Bhaddurgatte RC, Vijaya Kumar BP, Kusuma SM. Machine learning and prediction-based resource management in IoT considering Qos. International Journal of Recent Technology and Engineering. 2019;8(2):687–694.

[13] Fernando H, Marshall J. What lies beneath: Material classification for autonomous excavators using proprioceptive force sensing and machine learning. Automation in Construction. 2020;119(June):103374. Available from: https://doi.org/10.1016/j.autcon.2020.103374.

[14] Yazdi M, Sarafrazi K. Automated segmentation of concrete images into microstructures: A comparative study. Computers and Concrete. 2014;14(3): 315–325.

[15] Penumuru DP, Muthuswamy S, Karumbu P. Identification and

classification of materials using machine vision and machine learning in the context of industry 4.0. Journal of Intelligent Manufacturing. 2020;31(5): 1229–1241. Available from: https://doi.org/10.1007/s10845-019-01508-6.

[16] Hamledari H, McCabe B, Davari S. Automated computer vision-based detection of components of under-construction indoor partitions. Automation in Construction. 2017;74: 78–94.

[17] Yang J, Shi ZK, Wu ZY. Towards automatic generation of as-built BIM: 3D building facade modeling and material recognition from images. International Journal of Automation and Computing. 2016;13(4):338–349.

[18] Zhu Z, Brilakis I. Parameter optimization for automated concrete detection in image data. Automation in Construction. 2010;19(7):944–953. Available from: http://dx.doi.org/10.1016/j.autcon.2010.06.008.

[19] Araújo M, Martínez J, Ordóñez C, Vilán JAV. Identification of granite varieties from colour spectrum data. Sensors. 2010;10(9):8572–8584.

[20] Son H, Kim C, Kim C. Automated Color Model–Based Concrete Detection in Construction-Site Images by Using Machine Learning Algorithms. Journal of Computing in Civil Engineering. 2012;26(3):421–433.

[21] Braun A, Borrmann A. Combining inverse photogrammetry and BIM for automated labeling of construction site images for machine learning. Automation in Construction. 2019;106 (June):102879. Available from: https://doi.org/10.1016/j.autcon.2019.102879.

[22] Lee JH, Park SO. Machine learning-based automatic reinforcing bar image analysis system in the internet of things. Multimedia Tools and Applications. 2019;78(3):3171–3180.

[23] Ghassemi N, Mahami H, Darbandi MT, Shoeibi A, Hussain S, Nasirzadeh F, et al. Material Recognition for Automated Progress Monitoring using Deep Learning Methods. Journal of Advanced Engineering Informatics. 2020:1–30. Available from: http://arxiv.org/abs/2006.16344.

[24] Siddula M, Dai F, Ye Y, Fan J. Classifying construction site photos for roof detection. Construction Innovation. 2016;16(3):368–389.

[25] jrosebr1 (Adrian Rosebrock);. Available from: https://github.com/jrosebr1.

[26] Keras Tutorial: How to get started with Keras, Deep Learning, and Python - PyImageSearch;. Available from: https://www.pyimagesearch.com/2018/09/10/keras-tutorial-how-to-get-started-with-keras-deep-learning-and-python/.

[27] ralizadehsani/material_recognition;. Available from: https://github.com/ralizadehsani/material{_}recognition.

[28] Salah Alaloul W, Hannan Qureshi A. Data Processing Using Artificial Neural Networks. Dynamic Data Assimilation - Beating the Uncertainties [Working Title]. 2020 may. Available from: https://www.intechopen.com/online-first/data-processing-using-artificial-neural-networks.

Edited by Pier Luigi Mazzeo and Paolo Spagnolo

Deep learning is a branch of machine learning similar to artificial intelligence. The applications of deep learning vary from medical imaging to industrial quality checking, sports, and precision agriculture. This book is divided into two sections. The first section covers deep learning architectures and the second section describes the state of the art of applications based on deep learning.

Published in London, UK

IntechOpen

ISBN 978-1-83962-376-9
9 781839 623769