Digital Twin

A digital twin is an avatar of a real physical system which exists in the computer. The difference between computer model of a physical system and the digital twin is that digital twin tracks the temporal evolution of the physical system, they both attempt to closely match the behaviour of a physical system. [1]

Reifsnider and Mujumdar consider digital twin to be a high fidelity simulation integrated with an on-board health management system, maintenance history and historical vehicle and aircraft fleet data. [1]

Glaessgen and Stargel defined digital twin as an integrated multiscale, multiphysics probabilistic simulation of a complex product which uses the best available physical model, sensor updates etc. to mirror the life of the physical twin. [1]

Bosch defines digital twins as “connected devicesd such as tools, cars, machines, sensors, and other web-enabled thingsdin the cloud in a reusable and abstracted way. [2]

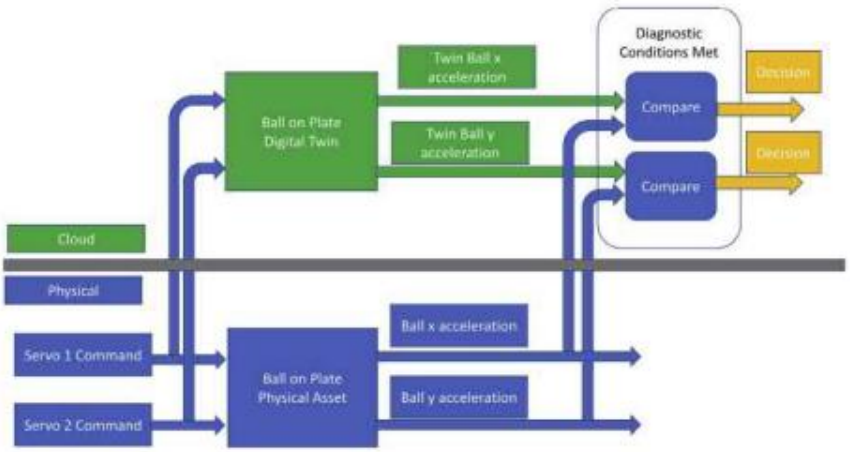
Examples – Simscape

Digital twin model creation of a robotic arm [2]

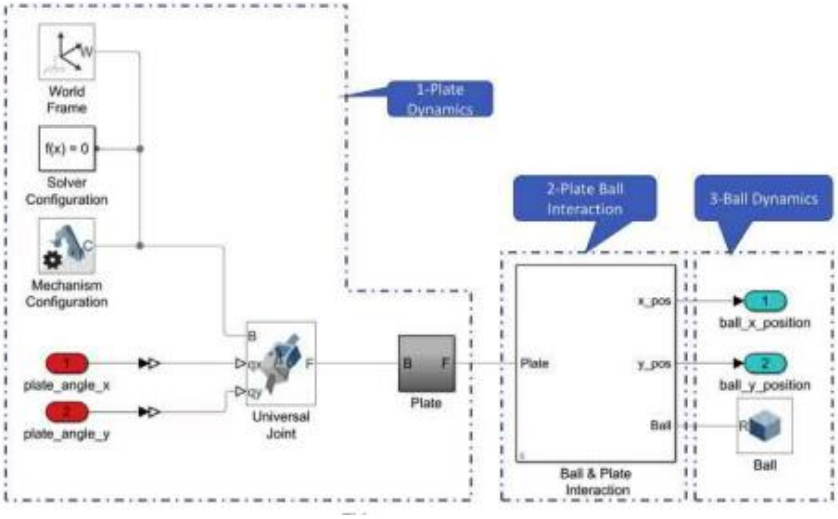
- Actuators to move different axis – specific x, y, z position
- 3 DoF
- Creation of model in Simcape using Multibody - Revolute joints, Translations …
- Sensing is very useful

Digital twin model creation of ball on plate [2]

- Two servo motors changing angle of the plate
- Goal : Keep the ball in the center of the plate
- Failure nodes: wiring issue, aging servo, mechanical failure in rotating mechanism, stuck ball

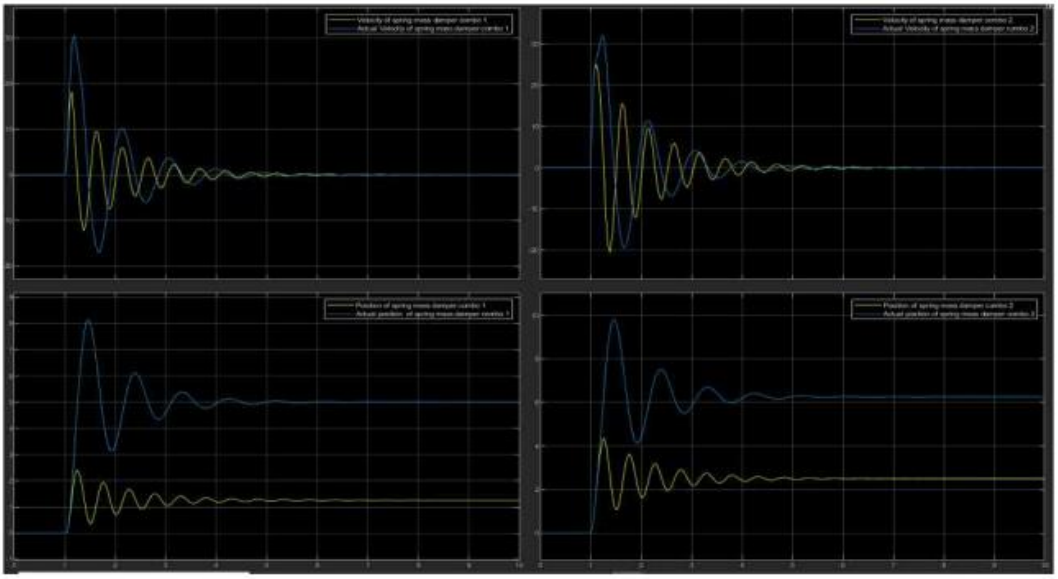


- Creation of model using Simscape Multibody with two PID controllers



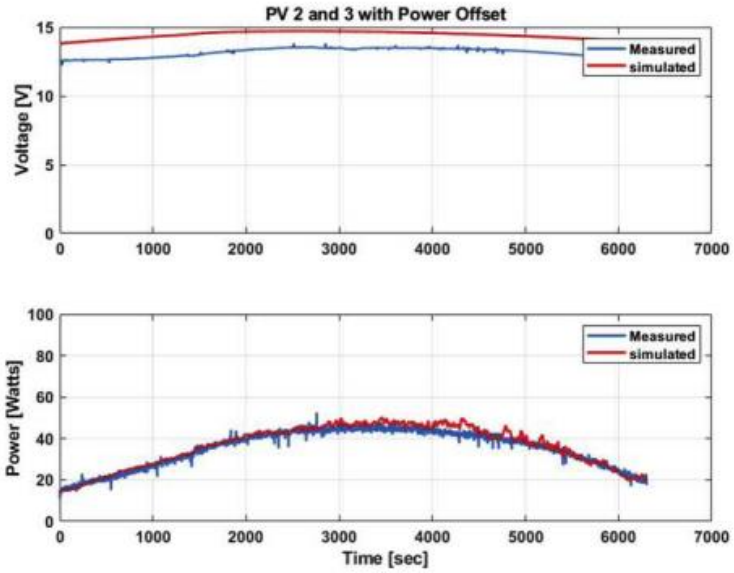
Digital twin model creation of double mass spring damper system [2]

- A spring, a damper and a mass



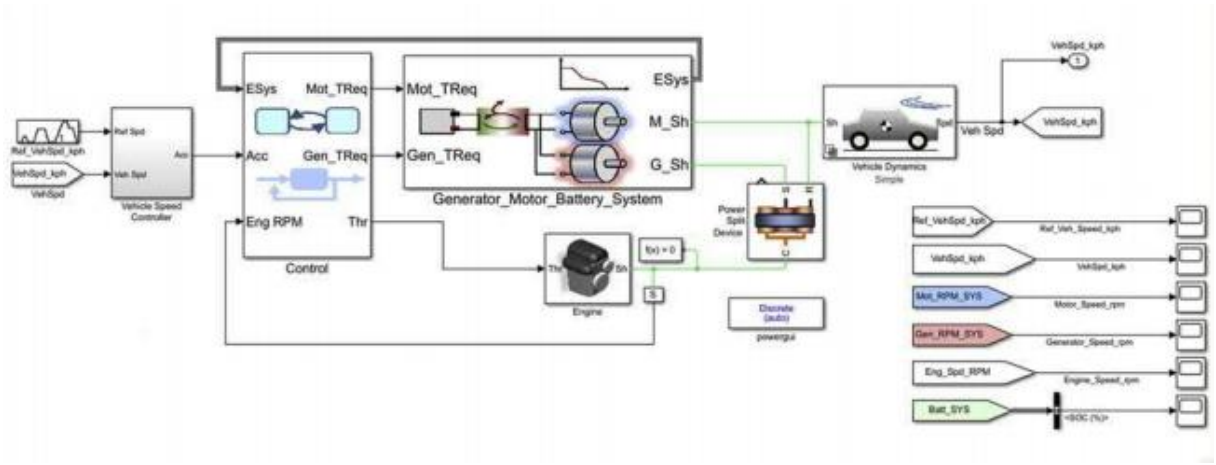
Physical asset – blue, Digital twin - yellow

Digital twin model creation of solar panels [2]

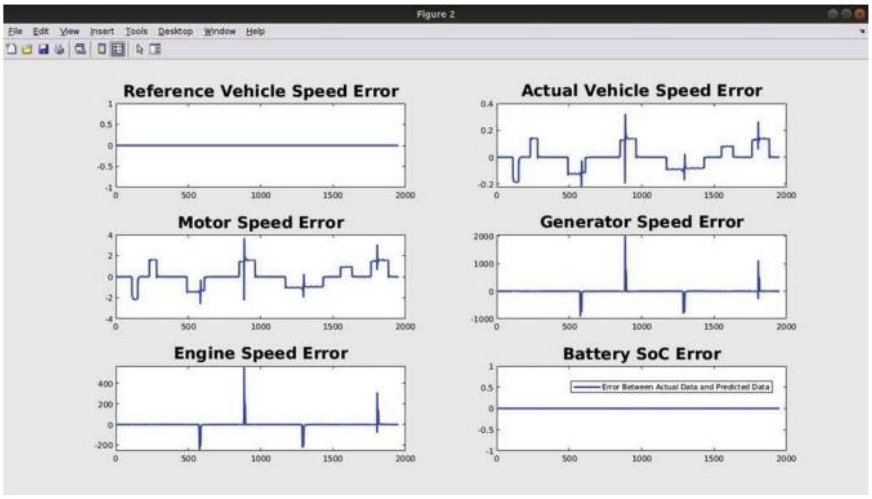


Digital twin development and cloud deployment for a Hybrid Electric Vehicle [2]

- Matlab, Simulink, Simscape into Raspberry Pi hardware board in real time
- Outputs – Actual Vehicle Speed, Motor Speed, Generator Speed, Engine Speed, Battery SoC



- Using Amazon Web Servis + Python



Machine Learning [3]

Supervised learning

- Algorithm that use previously-labeled data to learn its features.
- Classification or regression
- Classification: Email – SPAM or not SPAM (2 classes) – specific words (meeting, business, …)
- Regression: Number years the person is expected to live

Linear and logistic regression

- Goal: Minimalize a cost function by finding parameters
- Classification and Regression
- MSE (Mean square error) – popular cost function – square of difference between expected and predicted result
- Initialize the vector w – random values
- Use gradient descent to update the weights until MSE falls below threshold

1) We need to do an iteration between training data to calculate MSE (cost function)

2) We use algorithm gradient descent to actualize weights, then we need to calculate derivatives of the cost function to each weight

3) Cost function will increase or decrease, then we update weights

Support vector machines

- Mainly classification
- Tries to find hyperplane (a plane in a high-dimensional space), which separates the samples in the dataset (maximizes the distance between itself and points)
- For not lineary-separable data we can use soft margins or kernel trick (adding more dimensions)

Decision Trees

- Creates a classifier in the form of a tree (attributes)

Naïve Bayes

- Calculate the probability

Unsupervised learning

- We don’t use data beforehand, we let algorithm come to its conclusion
- Algorithm classify our data into clusters, in advanced algorithm you don’t have to specify the number of clusters

K-means

- Clustering algorithm

1) Choose k random points (centroids), which will represent the center of each of the k cluster
2) Assign each point with the closest centroid
3) For each cluster, we calculate new centroids by taking mean values of all points in the cluster
4) With new centroids we repeat 2), 3) until the stopping criteria is met

- Example: 4 franchises in the city (delivery locations)

Reinforcement learning

- Teaching machines how to play games
- Agent (machine) interacts with the environment, agent takes actions that can change the environment´s state (reward signals to decide its next action)

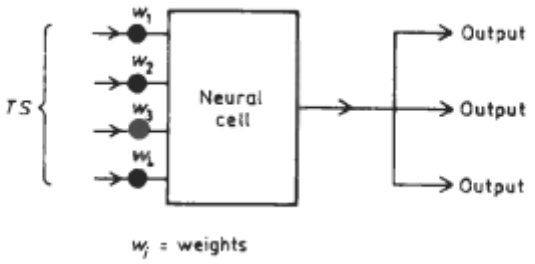
Q-learning

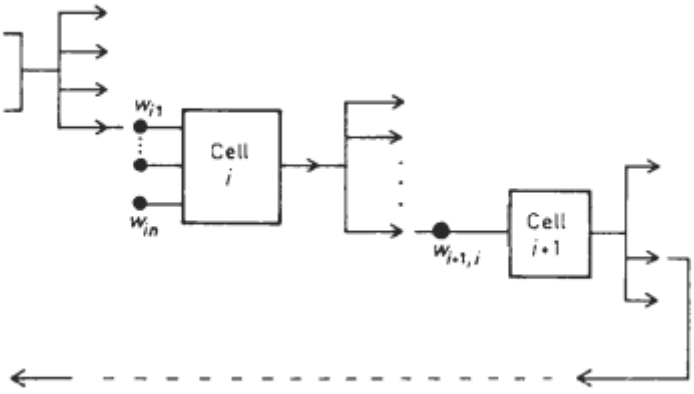
- An episode starts with a random initial state and finishes when we reach the terminal state
- Q-value in Q-table – higher, the more attractive the action is
- Example: Chess, but we can´t fit the table in memory, so we use neural network

Artificial neural networks

Artificial neural networks are, as their name indicates, computational networks which attempt to simulate, in a gross manner, the networks of nerve cell (neurons) of the biological (human or animal) central nervous system. This simulation is a gross cell-by-cell (neuron-by-neuron, element-by-element) simulation. It borrows from the neurophysiological knowledge of biological neurons and of networks of such biological neurons. It thus differs from conventional (digital or analog) computing machines that serve to replace, enhance or speed-up human brain computation without regard to organization of the computing elements and of their networking. [4]

Biological neural network [4]

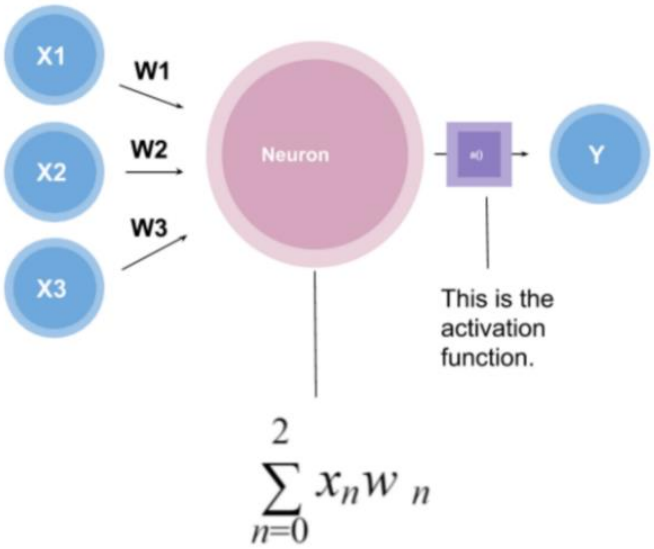




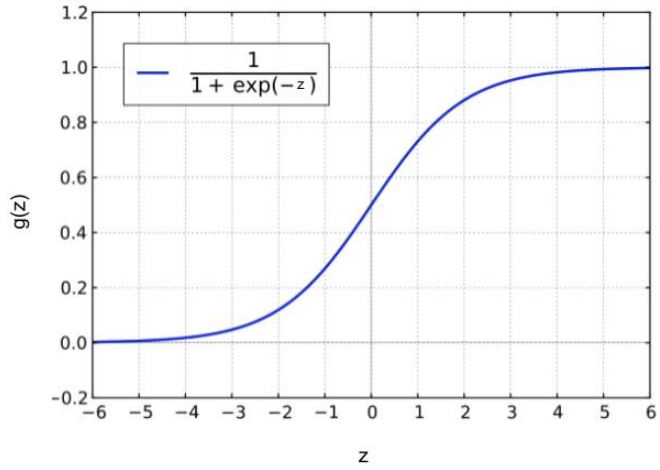
Basic Principles of ANNs [4]

- The activity of a neuron (ANN) is all-or-nothing.
- A certain fixed number of synapses larger than 1 must be excited within a given interval of neural addition for a neuron to be excited.
- The only significant delay within the neural system is the synaptic delay.
- The activity of any inhibitory synapse absolutely prevents the excitation of the neuron at that time.
- The structure of the interconnection network does not change over time

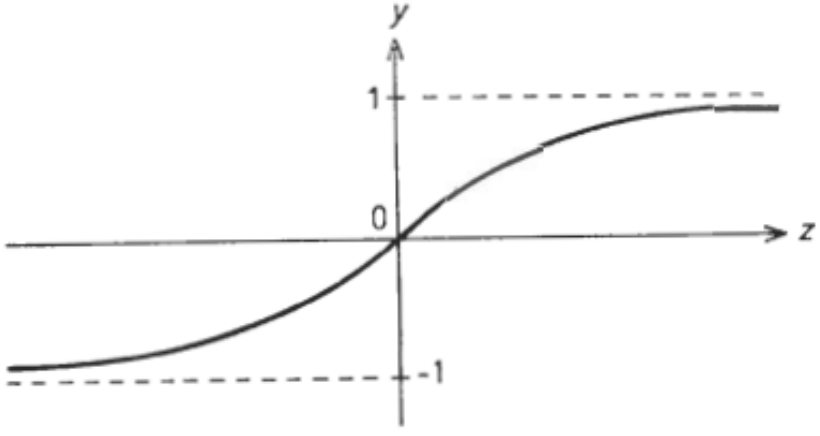
The Perceptron [5]



- Activation functions
    1. Unipolar - Logistic Function (Output 0 – 1)

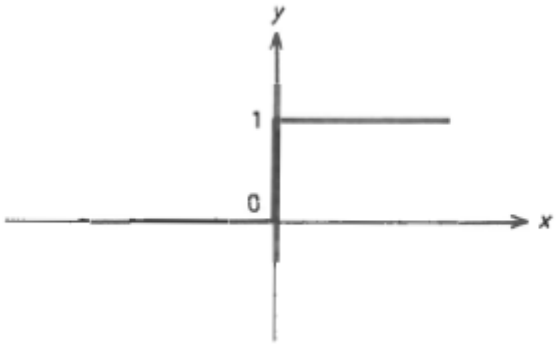


    2. Unipolar - Hyperbolic Tangent (Output -1 – 1) [4]

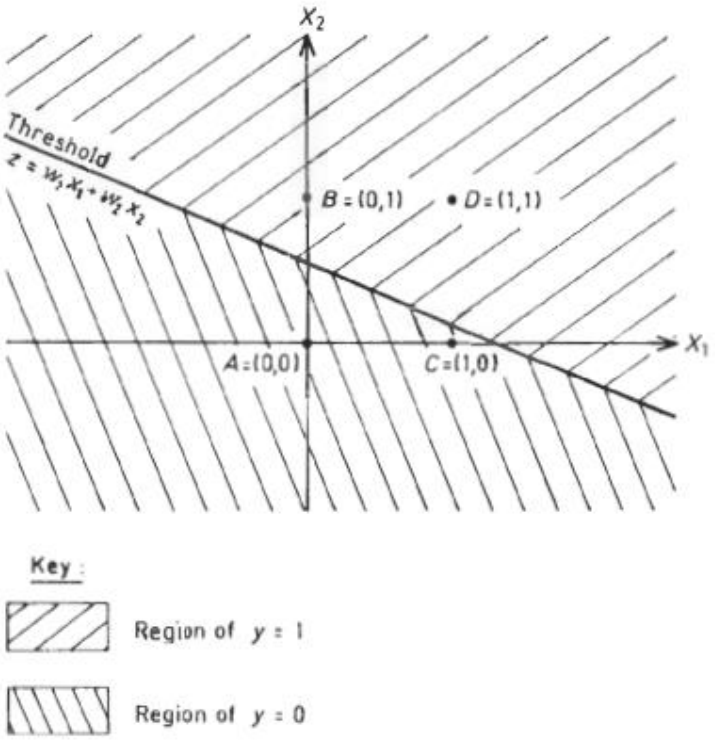


(a) $y = \frac{2}{1+\exp(-z)} - 1$

3. Binary activation function [4]



- Single layer percepton – 2 inputs and its output [4]



- Training

$$\hat{x}(n) = \sum_{i=1}^{M} \hat{a}_i x(n-i)$$

,

where $x\hat{}(n)$ is the estimate of $x(n)$.

$$e(n) \triangleq x(n) - \hat{x}(n)$$

,

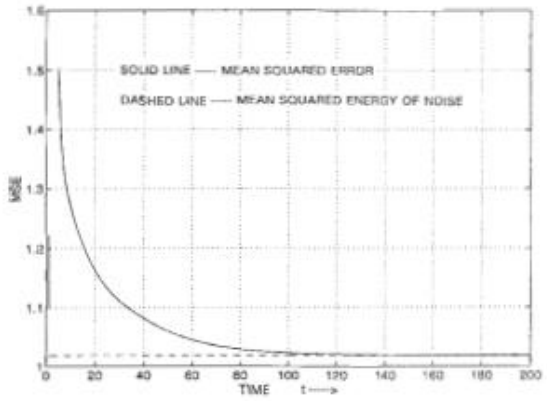
where e(n) is an error. Then

$$e(n) \to w(n)$$

Cost function - Mean square error

$$MSE \triangleq \hat{E}[e^2(n)] = \frac{1}{N}\sum_{i=1}^{N} e^2(i)$$

MSE vs. time

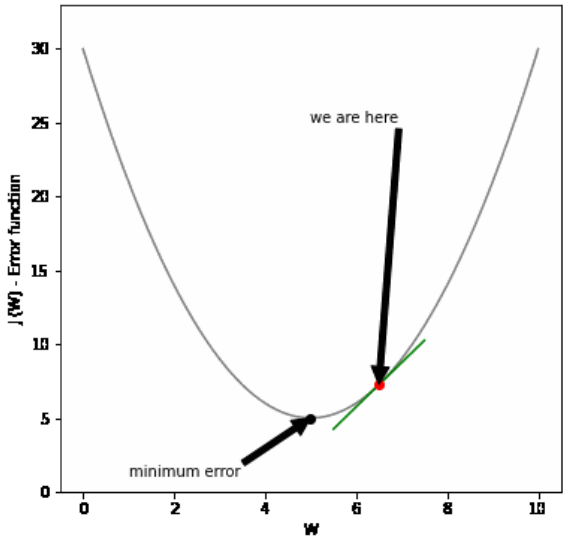


The Adaline – adaptive linear unit [6]

$$\begin{bmatrix} z_0 \\ \vdots \\ z_{n-1} \end{bmatrix} = \begin{bmatrix} w_{0,0} & \dots & w_{0,n-1} \\ \vdots & \ddots & \vdots \\ w_{m-1,0} & \dots & w_{m-1,n-1} \end{bmatrix}^T \cdot \begin{bmatrix} x_0 \\ \vdots \\ x_{m-1} \end{bmatrix} + \begin{bmatrix} b_0 \\ \vdots \\ b_{n-1} \end{bmatrix}$$

$$\begin{bmatrix} y_0 \\ \vdots \\ y_{n-1} \end{bmatrix} = f\left(\begin{bmatrix} z_0 \\ \vdots \\ z_{n-1} \end{bmatrix}\right)$$

- Training - Gradient Descent

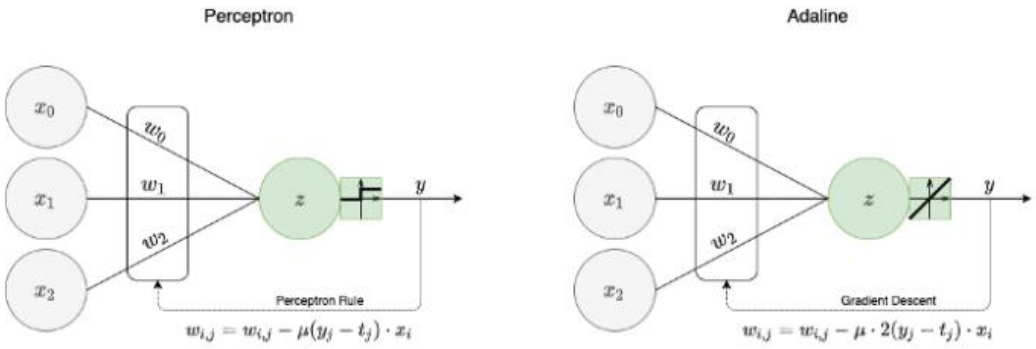


$$w = w - \mu \cdot \frac{\partial}{\partial w} J(w)$$

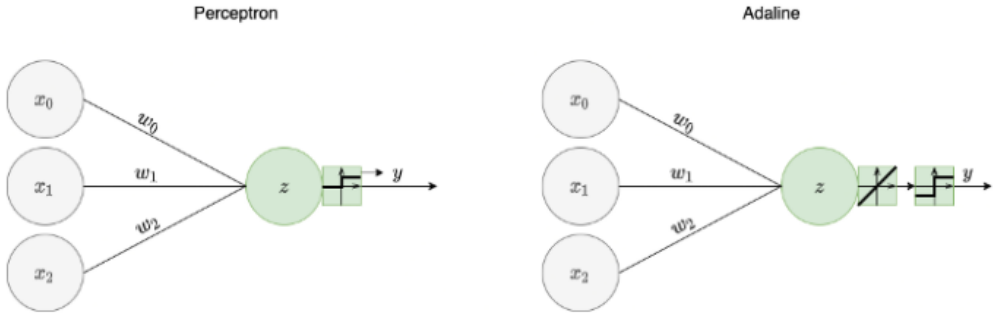
,

where μ is learning rate coefficient.

- Training phase

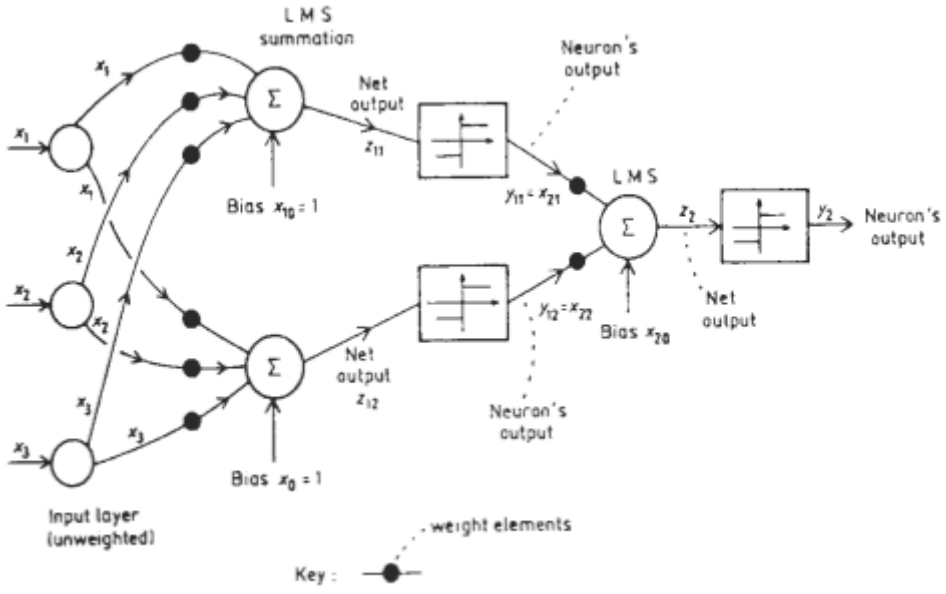


- Prediction phase

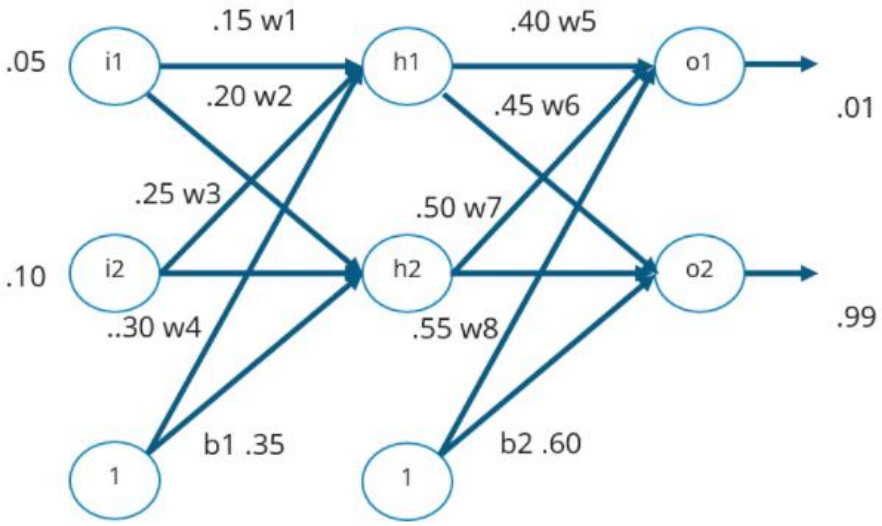


The Madaline – a multilayer extension of the single-neuron bipolar Adaline

- Madaline training differs from Adaline training in that no partial desired outputs of the inside layers are or can be available.
- Inside layers are called hidden layers.
- Training procedure is called Madaline Rule II. - Detailed in the publication [4]



Back Propagation [7]



1. Forward Propagation

$$\text{net h1} = w1*i1 + w2*i2 + b1*1$$

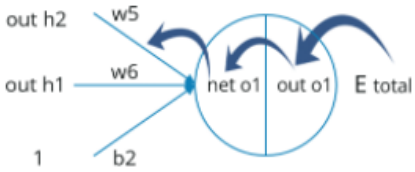
$$\text{out h1} = 1/1+e^{-\text{net h1}}$$

$$\text{E o1} = \Sigma 1/2(target - output)^2$$

$$E_{total} = \text{E o1} + \text{E o2}$$

2. Backward Propagation

$$\frac{\delta Etotal}{\delta w5} = \frac{\delta Etotal}{\delta out\ o1} * \frac{\delta out\ o1}{\delta net\ o1} * \frac{\delta net\ o1}{\delta w5}$$



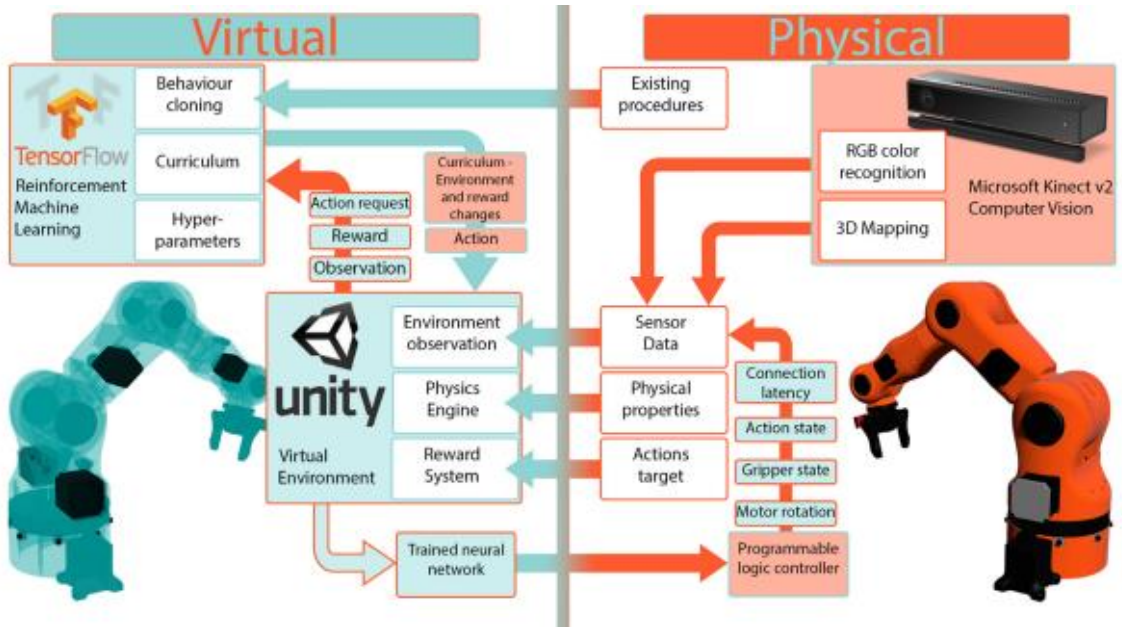
Training (Updating weights)

1. LMS
2. NLMS
3. NMLS
4. GNGD

Examples – Neural Networks

A robot arm digital twin utilising reinforcement learning [8]

- Virtual enviroment – Unity engine (Machine learning toolkit)
- Q-learning with convolutional neural networks (CNNs)



- Unity ml-agents – Agent (robot arm), Brain(makes the decisions), Academy(tracking and observing enviroment – send all data to TensorFlow)
- Agent need to choose the right action that maximalises the sum of rewards in the minimum number of time steps
- Proximal Policy Optimalization strategy to train policy
- Learning rate 10^{-4}, batch size of 128, maxSteps 10^{8}
- Ten cirriculum levels, red new task, action and new reward, blue carries on with current experiences (requires more accuracy or speed to receive the same reward)

Level	Curriculum adaptation
1	collision detection enabled - tolerance: 0.065
2	tolerance: 0.04
3	tolerance: 0.02
4	tolerance: 0
5	close gripper required for reward
6	carry cube to another free post for reward - tolerance: 0.4
7	tolerance: 0.2
8	tolerance: 0.1
9	tolerance: 0
10	open gripper to receive reward

Rewards and descriptions per curriculum levels.

Reward: Curriculum	Description
-1.f/`maxStep`: all	Reward added to incentivise agent exploration.
-0.05f: all	Added to disincentivise the robot to exceed any joint limits
-0.05f: all	Disincentivise actions to open gripper when it is open and vice versa
-1.0f: ≥ 1	Ends an episode if the arm collides with an object
+1.0f: < 5	For the gripper to reach the object
+1.0f: ≥ 5	For grabbing the cube within tolerance threshold
-0.1f: ≥ 5	Attempt to grab cube, but end affector too far
+1.0f: ≥ 6 & ≤ 9	Successfully moved a cube to another free post, within tolerances
-1.0f: ≥ 8	Ends an episode if a cube collides with a post to encourage lifting the end affector
+1.0f: 10	Cube grabbed, moved to another location and released with no mistakes
-1.0f: 10	Ends an episode if the cube is released at an incorrect location

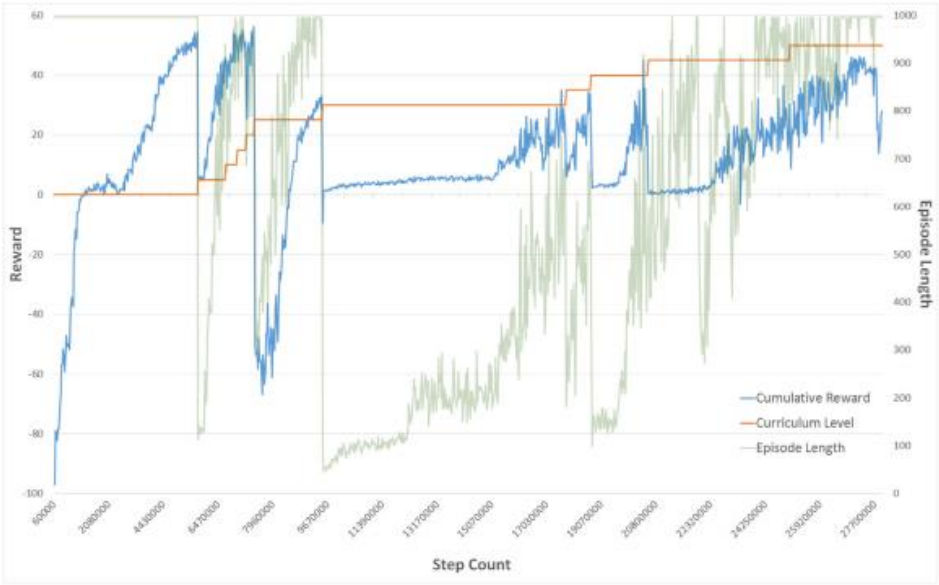
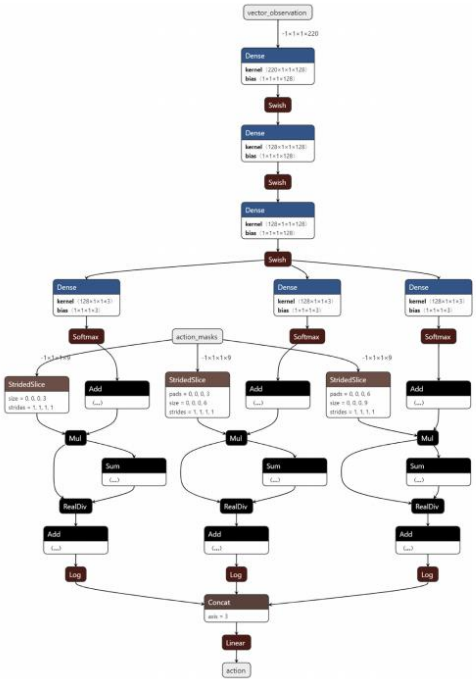


Fig. 8. Review of curriculum vs. reward and episode length in training.

- 30 hours to train



An artificial neural network model for predicting the performance of thermoacoustic refrigerators [9]

- ANN predicting the cooling temperature and performance
- Feedforward, 2 hidden layers (each 5 or 10 neurons) , 4 inputs – length, normalised stack position, stack porosity, frequency
- 280 experiments (randomly split into 70% for training sets, 15% for testing sets, 15% for validation sets)

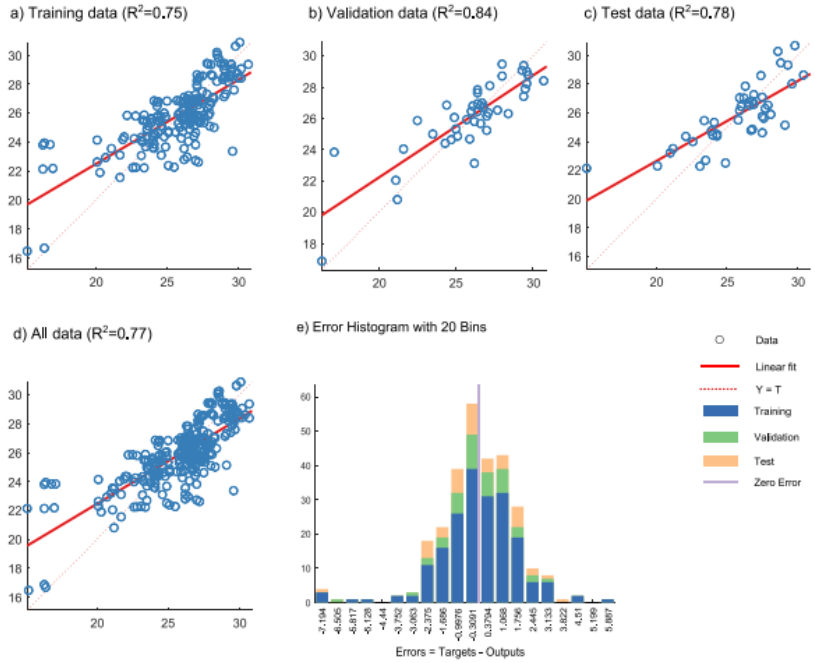


Fig. 4. The performance of the artificial neural network (ANN) using five neurons in the hidden layer. The square of the correlation between targeted values of the cooling temperature (on the x-axis) and predicted values of the cooling temperature (on the y-axis) in Celsius are shown for a) Training data b) Validation data c) Test data d) All data. The residuals for the training, validation and test data are shown in part (e). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

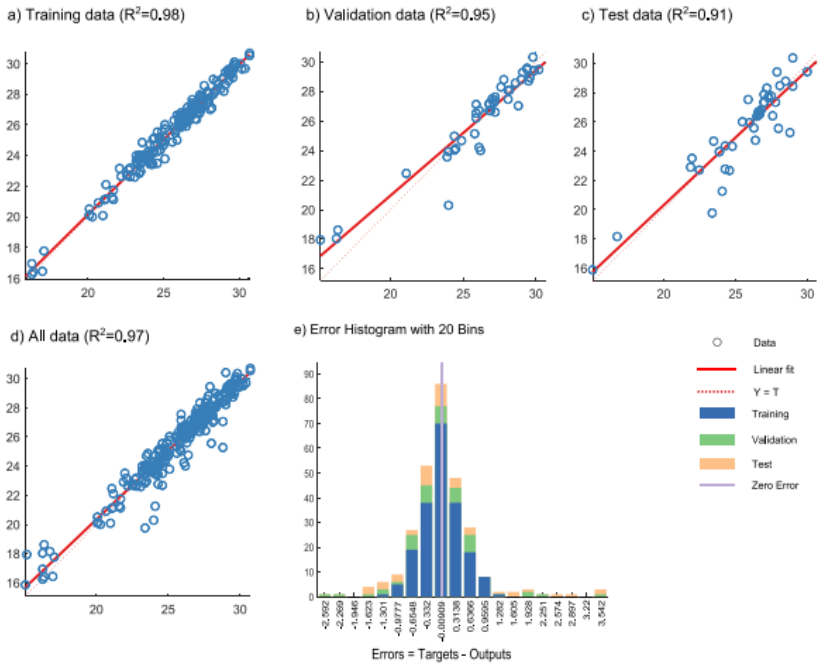
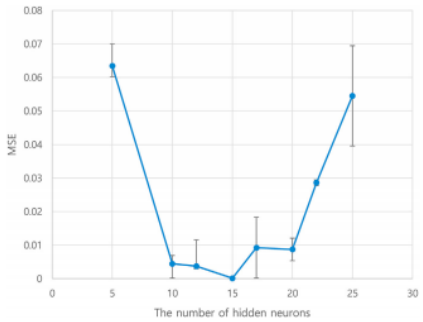


Fig. 5. The performance of the artificial neural network (ANN) using ten neurons in the hidden layer. The square of the correlation between targeted values of the cooling temperature (on the x-axis) and predicted values of the cooling temperature (on the y-axis) in Celsius are shown for a) Training data b) Validation data c) Test data d) All data. The residuals for the training, validation and test data are shown in part (e). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

- The effect of the number of neurons on model accuracy – model with 10 neurons had the highest R squared
- The effect of the number of hidden layers on model accuracy – could help to model complex non-linear relationships, but more layers lead to computational complexity

Development of model predictive control system using an artificial neural network: A case study with destillation column [10]

- ANN was adopted instead of using the existing linearized model in order to increase speed of optimization and accuracy of the model
- NARX model (MLP) – input layer, output layer and one or more hidden layers
- Levenberg-Marquardt, Bayesian Regularization – methods used to train NARX network (Backpropagation algorithm)
- In this case 3-layer feed-forward neural network was used (14 inputs (7 present, 7 previous), 15 hidden neurons, 2 outputs)
- MSE (square error) – between 5 and 25 hidden neurons
- 70% data to train, 15% to test, 15% for validation
- NNMPC – Neural network model predictive control



-

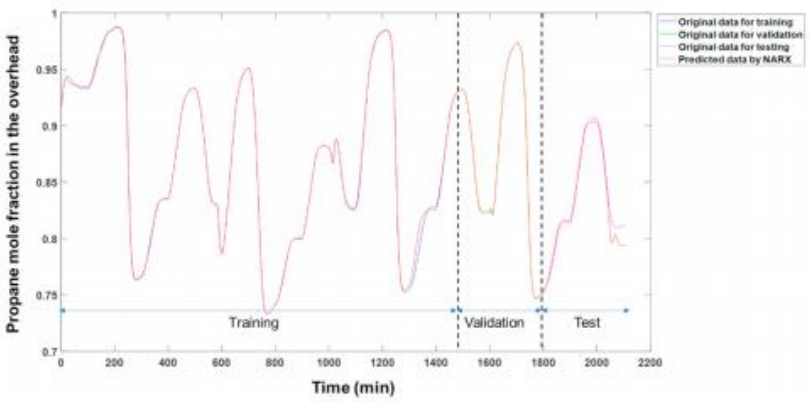
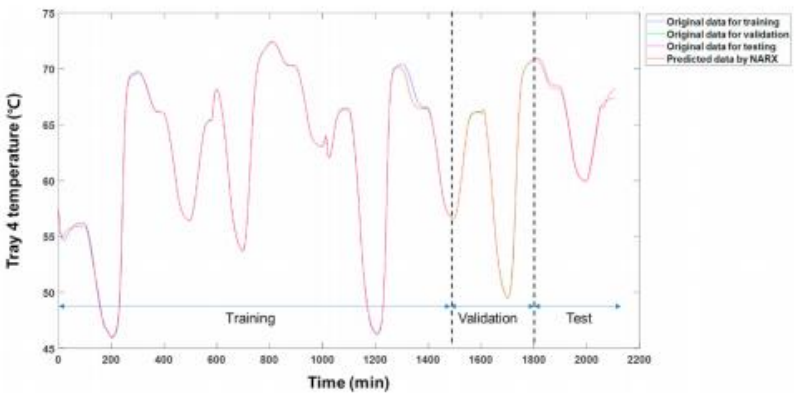


Fig. 10. Time response for propane mole fraction in the overhead.



-

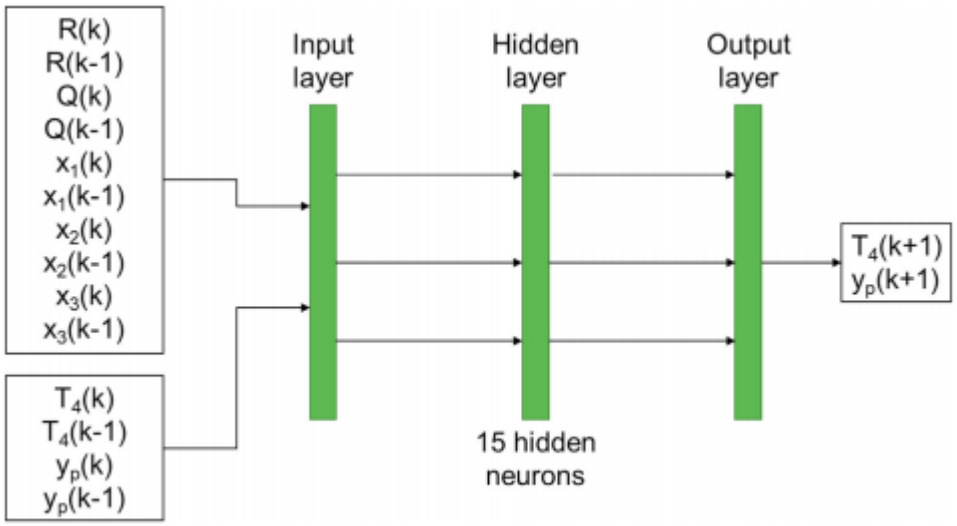


Fig. 8. Final NARX model architecture.

Digital Twin for 3D Printing on CNC Machines [11]

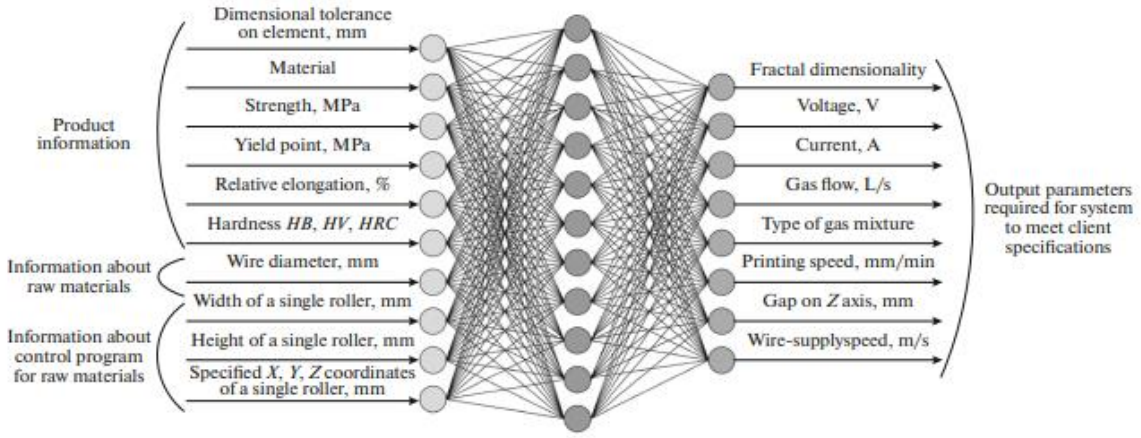
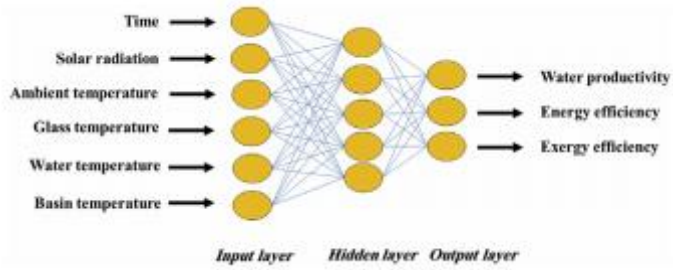


Fig. 5. Neural-network model (digital twin) of 3D printing on CNC machines.

A proper model to predict energy efficiency, exergy efficiency, and water productivity of a solar still via optimezed neural network [12]

- MLP neural network



- ICA approach

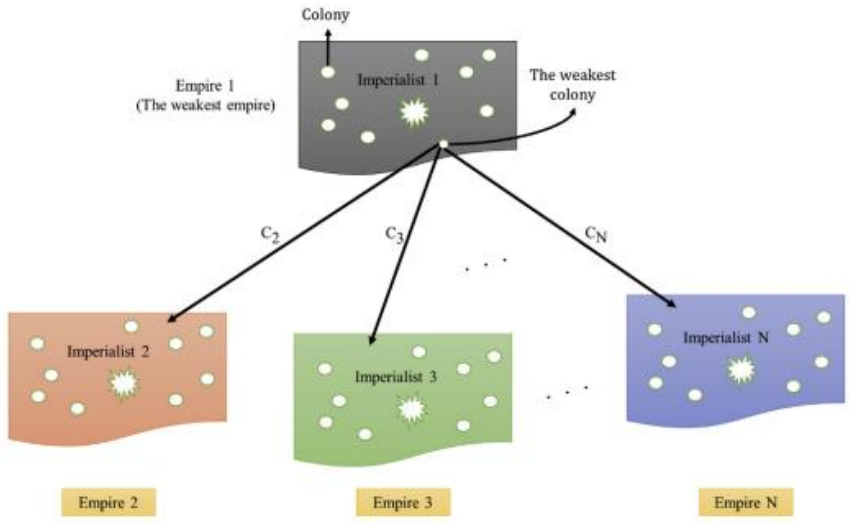


Fig. 3. A graphical statement of the ICA approach.

- 80% training, 20% testing
- $R^2 > 0.96$
- 6 hidden neurons

Adaptive Neural Predictive Control for Permanent Magnet Synchronous Motor Systems With Long Delay Time [13]

- Nonlinear multivariable time varying system and the existence of uncertainties and disturbances – difficult to control (PI control – loss of high performance)
- MPC – control of system with complex constraints, large delay time, non-minimum phase, or nonlinearity
- MPC – minimizing the designed cost function at each time step
- Network – Adaptive neural predictive control
- Most current predictive control methods are modified from generalized predictive control (GPC)
- MPC is composed of an optimization procedure and a prediction model (predictor)
- The future control sequence is achieved by minimizing a cost function
- Cost function :

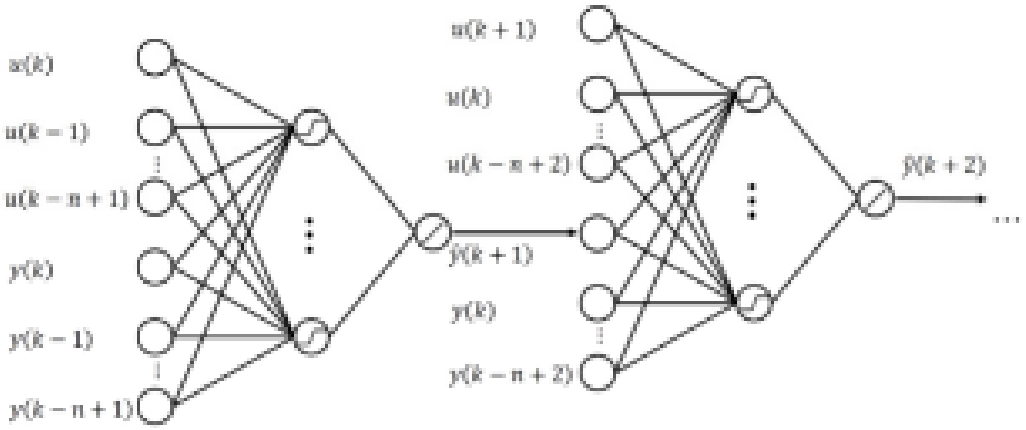
$$J = E + D$$
$$E = \sum_{j=N_1}^{N_2} \left[y_r (k+j) - \hat{y} (k+j) \right]^2$$
$$D = \sum_{j=1}^{N_2} \lambda_j \left[\Delta u (k+j-1) \right]^2.$$

  - N1 minimum prediction horizon, N2 maximum prediction horizon, yr(k+j) reference input, y^(k+j) predicted output, (delta)u(k+j-1) = u(k+j-1) – u(k+j-2), u(k+j-1) control signal, lambda weights the difference of control signals to gain the smother control sequence
  - Single hidden layer

$$\hat{y}(k+1) = \sum_{j=0}^{N_H-1} w_j h_j(k)$$

$$h_j(k) = \sigma\left[\sum_{i=0}^{n-1} a_{i,j} y(k-i) + b_{i,j} u(k-i)\right],$$

- Nh number of hidden neurons, hj output from the jth hidden neuron, wj, aij, bij weights on links, sigma sigmoid activation function



- Extended Kalman Filter (EKF) based learning algorithm approximately gives the minimum variance estimate of the weights in NN, fewer iterations than steepest descent based methods
- EKF used for system identification of the plant and model adaption
- The cost function:

$$J = \sum_{j=1}^{N} \left\{ \left[y_r(k+j) - \hat{y}(k+j) \right]^2 + \lambda \left[\Delta u(k+j-1) \right]^2 \right\}.$$

- The cost function for predictive speed controler based on ANPC:

same as ANPC, Fig. 2, where $y_r(k) \equiv \omega_r(k)$, $y(k) \equiv \omega(k)$ and $\hat{y}(k+1) \equiv \hat{\omega}(k+1)$. To make the controller more flexible and gain the better performance, the cost function can be rewritten as following:

$$J = \sum_{j=1}^{N} \left\{ \begin{array}{l} \left[\omega_r(k+j) - \hat{\omega}(k+j) \right]^2 \\ +\lambda_1 \left[\Delta u(k+j-1) \right]^2 + \lambda_2 \left[u(k+j-1) \right]^2 \end{array} \right\}, \quad (6)$$

The added constraint, $U_2 = \sum_{j=1}^{N} \lambda_2 \left[u(k+j-1) \right]^2$, represents that the excepted optimal control signal in the future should be as small as possible.

- The cost function for predictive position controller:
- To gain higher resoluton of rotation angle

$$J = \sum_{j=1}^{N} \left\{ \begin{array}{l} \left[\theta_r (k+j) - \hat{\theta}(k+j) - \hat{e}_\theta (k)\right]^2 \\ +\lambda_1 \left[\Delta u (k+j-1)\right]^2 \\ +\lambda_2 \left[u (k+j-1) - g_u \cdot e^{-\tau \cdot k}\right]^2 \\ +\lambda_3 \left[\hat{\omega}(k+j)\right]^2 \end{array} \right\}, \quad (7)$$

where g_u represents the lower bound that the control signal can overcome the maximum static friction, τ is utilized to dispel the constraint of g_u after the plant is working, and the constraint to get the slower speed is $\Omega = \sum_{j=1}^{N} \lambda_3 \left[\hat{\omega}(k+j)\right]^2$.

$$\theta (k+1) = \theta (k) + \frac{T_s}{2} \cdot (\omega (k+1) + \omega (k)),$$

- Fuzzy compensator
    - Problems in the practical system – backlash, friction, signal resolution, sampling period – possition cannot be well controlled
- NN predictor 6 inputs, 4 hidden neurons

Digital twin, physics-based model, and machine learning applied to damage detection in structures [14]

Another examples to detect damage:

M.K.D. Knezevic, K. Willcox, Toward predictive digital twins via component-based reduced-order models and interpretable machine learning, AIAA Scitech Forum 0418.

C. Bigoni, J.S. Hesthaven, Simulation-based anomaly detection and damage localization: an application to structural health monitoring, Comput. Meth. Appl. Mech. Eng. 363 (2020) 112896.

D. Alves, G. Daniel, H. de Castro, T. Machado, K. Cavalca, O. Gecgel, J. Dias, S. Ekwaro-Osire, Uncertainty quantification in deep convolutional neural network diagnostics of journal bearings with ovalization fault, Mech. Mach. Theory 149 (2020) 103835

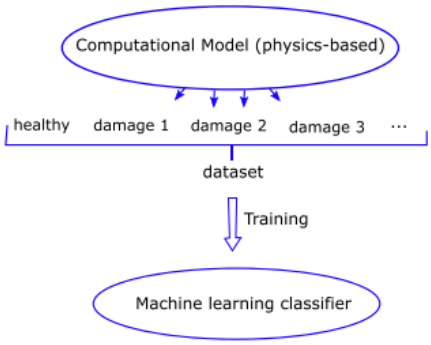


Fig. 6. The physics-based computational model is used to construct a dataset that is used to train a machine learning classifier.

Table 1
Accuracy of different classifiers.

Classifier	Accuracy
Quadratic Discriminant	93.3%
SVM (quadratic)	93.1%
SVM (linear)	92.2%
SVM (cubic)	86.0%
Linear Discriminant	84.8%
KNN	81.8%
SVM (Gaussian)	80.6%
Ensemble (Bagged Trees)	77.9%
Decision Tree	61.2%
Ensemble (RUSBoosted Trees)	39.5%

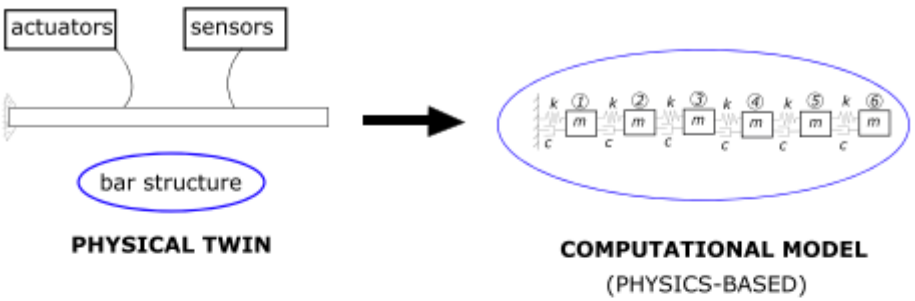


Fig. 2. Sketch of the physical twin and the corresponding physics-based computational model.

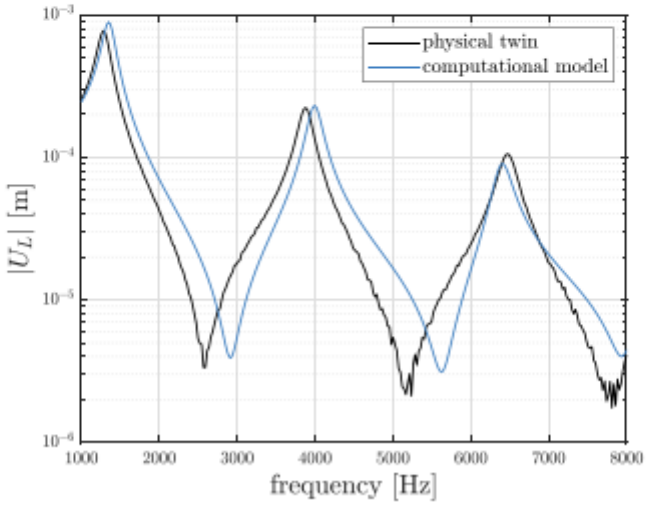
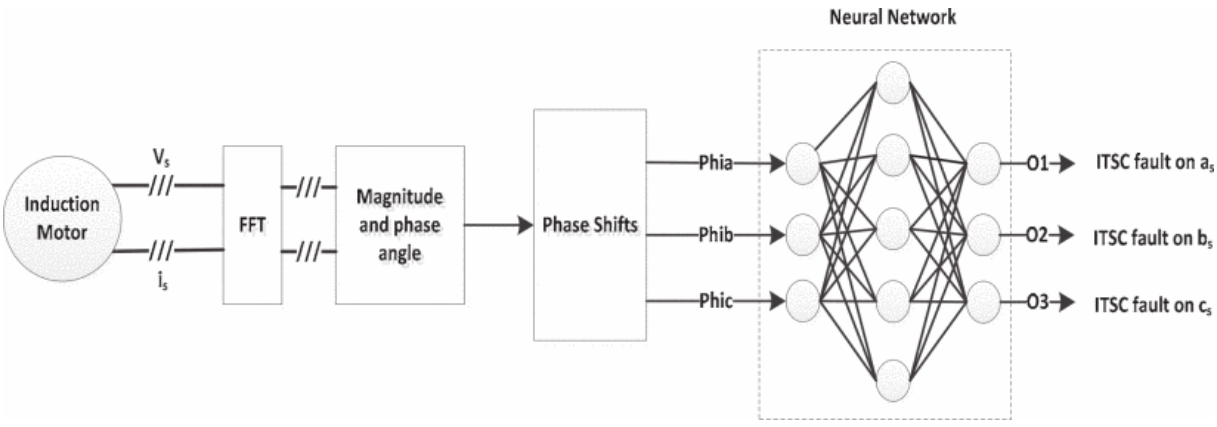


Fig. 3. Frequency responses of physical twin and the deterministic computational model.

Asynchronous motors fault detection using ANN and fuzzy logic methods [15]

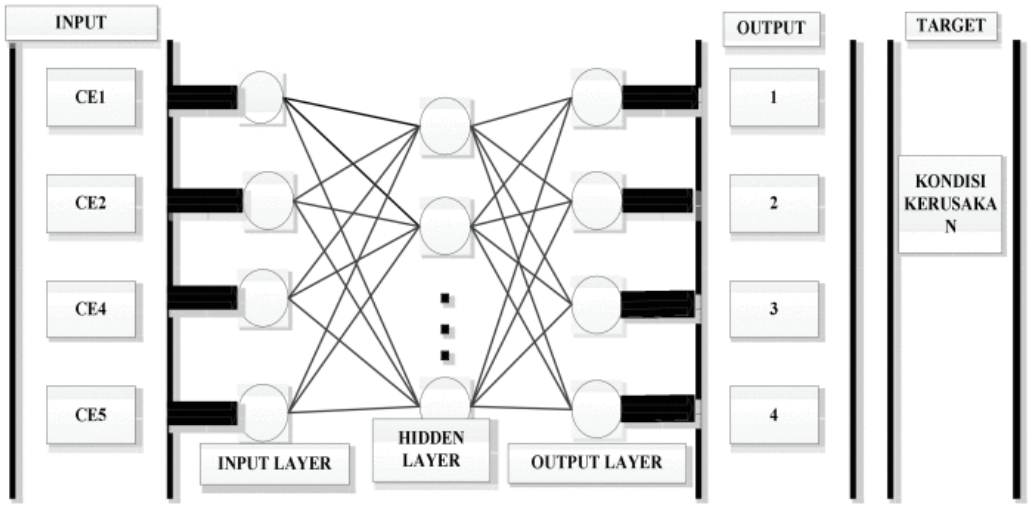
- A feedforward multi-layer perceptron (MLP) Neural Network trained by back propagation (BP) algorithm is used in order to automatically detect and locate ITSC fault.



Detection Of Induction Motor Bearing Damage With Starting Current Analysis Using Wavelet Discrete Transform And Artificial Neural Network [16]

- 100% for inner-race damage, 98% for outter-race damage and 100% for ball bearing damage
- artificial neural network used is backpropagation with one input layer, one hidden layer and one output layer
- training data used are 80% of the total 350 data, and 20% as test data

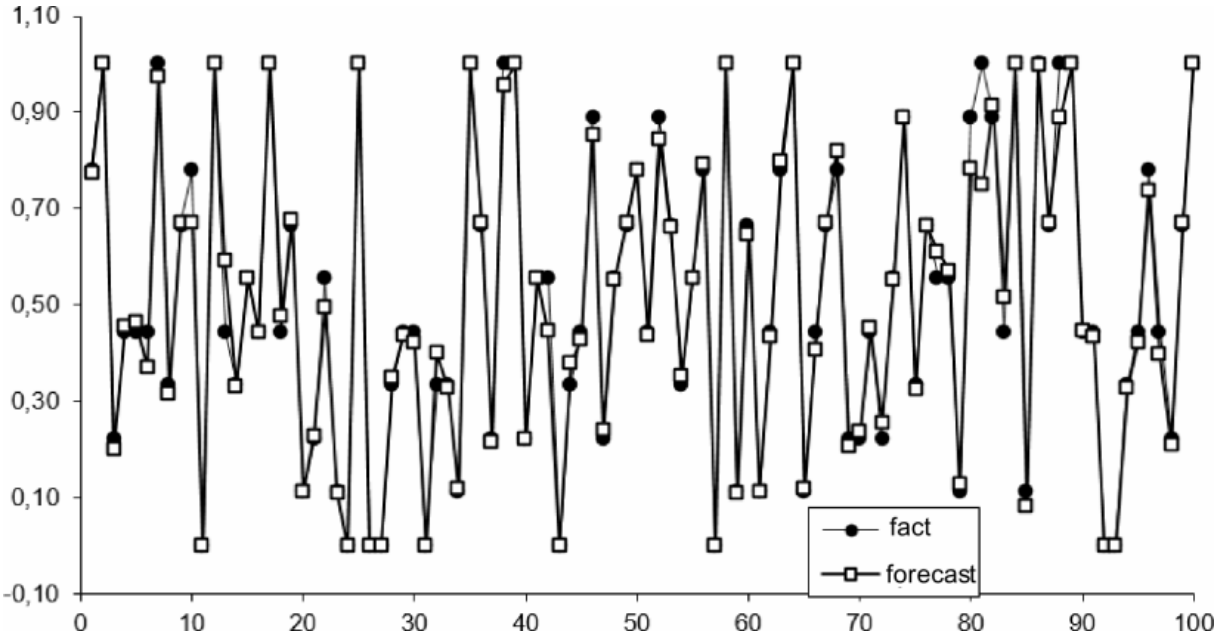
Case	Case variation	Load (%)	Total
Normal	-	0; 25; 50; 75; 100;	5
Inner-Race bearing damage	1mm; 2mm; 3mm; 4mm; 5mm.	0; 25; 50; 75; 100;	25
Outer-Race bearing damage	1mm; 2mm; 3mm; 4mm; 5mm.	0; 25; 50; 75; 100;	25
Ball bearing damage	1 ball; 2 ball; 3 ball	0; 25; 50; 75; 100;	15
Total			70



CE = Coefisien Energy, with normal data target is 1, inner race damage is worth 2, damage to outter race is 3 and defective bearing ball value is 4

Identification of the Asynchronous Electric Motor Defects Based on Neural Networks [17]

Three NN - 4-10-1, 17-25-15-1, 185 hidden or two layers 7 + 4 (better)



17-25-15-1

The learning speed in all experiments is taken as 0.01

To exclude overfitting of the neural network, the following recommendations are taken into consideration.

1. The number of neurons in the input and output layers is rigidly determined by the number of input and output variables of the model accordingly.

2. The number of neurons in the hidden layers and the number of hidden layers are chosen in such a way that the number of formed links is at least two to three times less than the number of training examples.

References

[1] R. Ganguli, S. Adhikari, The digital twin of discrete dynamic systems: Initial approaches and future challenges, Applied Mathematical Modelling, Volume 77, Part 2, 2020, Pages 1110-1128, ISSN 0307-904X, https://doi.org/10.1016/j.apm.2019.09.036

[2] KHALED, Nassim, PATTEL, Bibin a SIDDIQUI, Affan. Digital Twin Development and Deployment on the Cloud. USA: Mara Conner, 2020. ISBN 978-0-12-821631-6.

[3] IVAN, Vasilev. Python Deep Learning. 2nd Edition. UK: Packt Publishing, 2019. ISBN 978-1-78934-846-0.

[4] GRAUPE, Daniel. PRINCIPLES OF ARTIFICIAL NEURAL NETWORKS. 2nd Edition. USA: World Scientific Publishing Co. Pte., 2007. ISBN 978-981-270-624-9.

[5] Anjali Bhardwaj, What is a Perceptron? – Basics of Neural Networks, 2020, What is a Perceptron? – Basics of Neural Networks | by Anjali Bhardwaj | Towards Data Science

[6] Adriano Vinhas, Adaline neural networks: the origins of gradient descent, 2021, Adaline neural networks: the origins of gradient descent | by Adriano Vinhas | Towards Data Science

[7] Saurabh, Backpropagation – Algorithm For Training A Neural Network, 2020, What Is Backpropagation? | Training A Neural Network | Edureka

[8] Marius Matulis, Carlo Harveya, A robot arm digital twin utilising reinforcement learning, 2021

[9] Mahmoud A. Alamir, An artificial neural network model for predicting the performance of thermoacoustic refrigerators, 2020

[10] Yeonju Shin, Robin Smith, Sungwon Hwang, Development of model predictive control system using an artificial neural network: A case study with a distillation column, 2020

[11] Yu. G. Kabaldina, P. V. Kolchina, D. A. Shatagina, M. S. Anosova, and A. A. Chursinb, Digital Twin for 3D Printing on CNC Machines, 2019

[12] Saeed Nazari, Mehdi Bahiraei, Hossein Moayedi, Habibollah Safarzadeh, A proper model to predict energy efficiency, exergy efficiency, and water productivity of a solar still via optimized neural network, 2020

[13] B. -F. Wu and C. -H. Lin, "Adaptive Neural Predictive Control for Permanent Magnet Synchronous Motor Systems With Long Delay Time," in *IEEE Access*, vol. 7, pp. 108061-108069, 2019, doi: 10.1109/ACCESS.2019.2932746.

[14] T.G. Ritto, F.A. Rochinha, Digital twin, physics-based model, and machine learning applied to damage detection in structures, 2021

[15] N. Lashkari, H. F. Azgomi, J. Poshtan and M. Poshtan, "Asynchronous motors fault detection using ANN and fuzzy logic methods," *2016 IEEE Energy Conversion Congress and Exposition (ECCE)*, 2016, pp. 1-5, doi: 10.1109/ECCE.2016.7854890.

[16] E. Navasari, D. A. Asfani and M. Y. Negara, "Detection Of Induction Motor Bearing Damage With Starting Current Analysis Using Wavelet Discrete Transform And Artificial Neural Network," *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE)*, 2018, pp. 316-319, doi: 10.1109/ICITEED.2018.8534749.

[17] S. Dmitry, S. Maxim and Z. Dmitry, "Identification of the Asynchronous Electric Motor Defects Based on Neural Networks," *2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT)*, 2017, pp. 1-4, doi: 10.1109/ICAICT.2017.8686992