sciendo Journal of Electrical Engineering, Vol. 76, No. 1, 2025, pp. 99-105

Overview of implementation principles of artificial intelligence methods in industrial control systems

Ladislav Körösi, Slavomír Kajan

The proposed article discusses the principles of implementing artificial intelligence (AI) methods in industrial control systems focusing on the deployment of neural networks (NNs) within programmable logic controllers (PLCs). Recent advancements in AI have led to significant improvements in modeling, control, quality control and predictive maintenance. This progress is further supported by the development of newer CPU architectures in PLCs, which offer enhanced processing speeds and capabilities. These advanced CPUs facilitate the implementation of more complex AI algorithms enabling systems to perform real-time data analysis. By leveraging the power of AI and improved hardware, industries can achieve higher levels of automation and better decision-making.

Keywords: PLC, NN, modelling, control, real-time

1 Introduction

In recent years, AI has experienced remarkable progress across various fields including large language models (LLMs), convolutional neural networks (CNNs), generative adversarial networks (GANs) and others. These technologies are being progressively incorporated into sophisticated control and modeling solutions for automation, quality assurance, predictive maintenance, human machine interactions (HMI) and various other applications.

Neural networks can be deployed directly within PLCs or leveraged on more powerful external systems such as PCs, industrial PCs and edge devices. These devices then act as master systems to process data and transmit the results of the processing to the control system. The direct implementation of neural networks in PLCs poses distinct challenges, primarily stemming from the high computational requirements of these networks, which can impact their ability to process data in real time. The integration of neural networks into PLC's CPU offers substantial potential for improving functionality and performance across a variety of applications, despite certain challenges [1-8]. Our earlier research showed that although CNNs could be applied for gesture recognition directly in PLCs CPU, realizing real-time control continued to pose a challenge [4].

In parallel, edge devices and other powerful master systems enable the processing of complex algorithms, allowing the inference in real-time. Numerous studies highlight the potential of machine vision-based image processing in industrial and health applications, including object identification, quality assurance, disease detection, human activity and gesture recognition [6, 9-13]. For example, advancements in machine learning-based neural networks, such as multilayer perceptrons (MLPs) and long short-term memory networks (LSTMs) continue to be refined for real-time control applications [14-16].

This paper aims to provide an overview of the implementation principles of AI methods in industrial control systems, specifically focusing on the significance of deploying NNs within PLCs while recognizing the role of master systems. By analyzing the properties of PLCs we seek to demonstrate how the integration of AI can enhance the performance and reliability of industrial control systems through efficient processing and decision-making capabilities. In our overview we will show examples from Siemens and Schneider Electric control systems.

2 Variables

The variable (often referred to as a tag) in PLC systems [17-20] is defined by at least a data type and a physical address (such as %I for inputs, %Q for outputs and %M for states) or by a data type and a symbolic name. Nowadays, modern CPUs support both approaches, however symbolic programming is more commonly preferred. This section will focus on variables belonging to either the elementary data type group or the user-defined data type group for storing NNs parameters and process inputs and outputs of the technology.

Institute of Robotics and Cybernetics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, Bratislava, Slovakia
ladislav.korosi@stuba.sk, slavomir.kajan@stuba.sk

https://doi.org/10.2478/jee-2025-0010, Print (till 2015) ISSN 1335-3632, On-line ISSN 1339-309X

2.1 Elementary variables

Elementary variables (EVs) are those variables that belong to the data types such as INT (16 bit), DINT (32 bit), REAL (32 bit), LREAL (64 bit), BOOL (1 bit) and so on [17-24]. To store NN parameters depending on the precision needed and the data types supported by the CPU, REAL or LREAL are the recommended options. The disadvantage of EVs is that they must be created one by one, which is not effective for larger NN structures. An example of elementary variables (EVs) is shown in Fig. 1. The first two examples are for Siemens (TIA Portal) representing different memory areas (Data Block – it will be discussed later), while the third entry is for Schneider Electric (Unity Pro; nowadays EcoStruxure Control Expert).

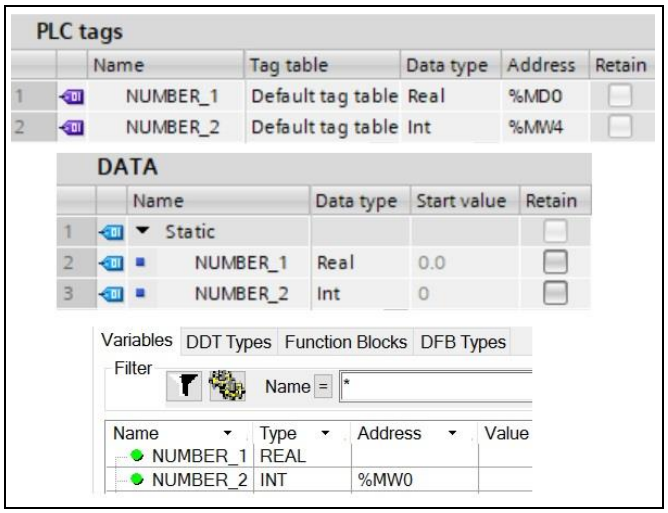


Fig. 1. Examples of elementary variables

2.2 Data blocks and structures

An effective way to define variables is by using DBs in Siemens PLCs or by structured variables in Schneider Electric PLCs. These concepts are essentially the same but are referred to differently by each manufacturer. By using ARRAY one can create one or more dimensional variables, which are widely used to store weights, biases and other related data. An example of a two-dimensional variable W1 is shown in Fig. 2. There are two crucial properties of this variable: the start value (initial value) and retention. The start values can be used for importing trained neural network parameters, while retention preserves the current values during a power loss of the CPU for example during training procedure.

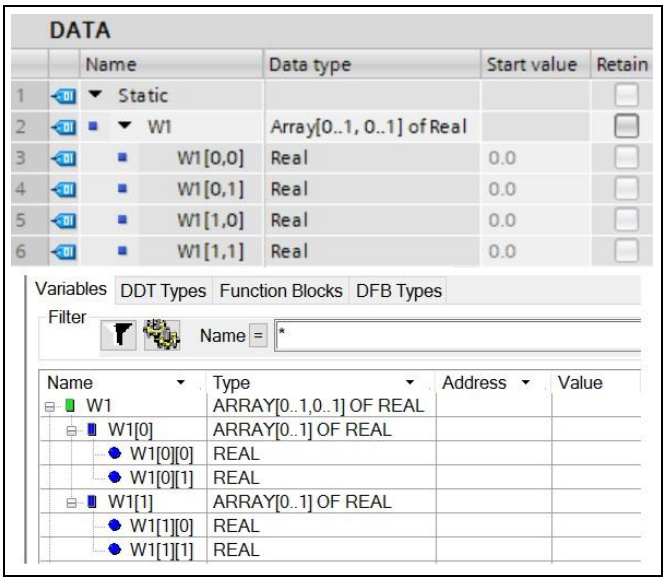


Fig. 2. Examples of array variables

A more effective approach is to define User Defined Data Types (UDTs) in Siemens [18-19] terminology or Derived Data Types (DDTs) in Schneider Electric terminology [17] combined with ARRAYS. A simple example in TIA Portal is shown in Fig. 3. The upper part defines the UDT, while the lower part displays the structured variable.

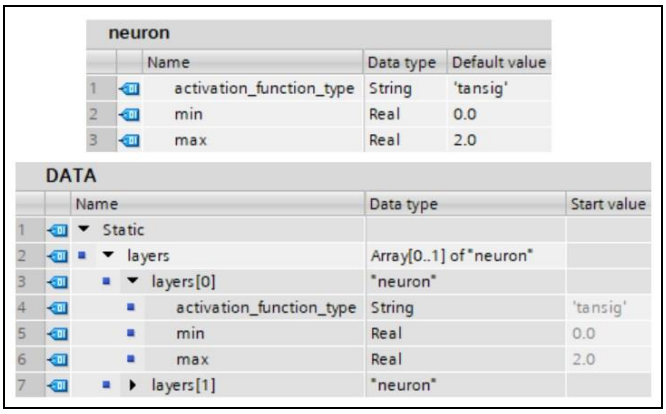


Fig. 3. Example of UDT and structured variable

3 Programming language

IEC 61131-3 defines five programming languages for PLCs [17-21, 23, 24]:

- Ladder Diagram (LD) – A graphical programming language resembling electrical relay logic.
- Function Block Diagram (FBD) – Another graphical programming language. It uses function blocks (FB) for creating complex functions and control logic.

- Structured Text (ST) – A high-level text-based language similar to Pascal.
- Instruction List (IL) – A low-level assembly-like language for sequential operations.
- Sequential Function Chart (SFC) – A graphical language for programming complex sequences with steps and transitions similar to a flowchart.

Structured Text is ideal for programming NNs within PLC environments due to its readability, support for complex data types, strong mathematical capabilities and fast software development.

4 Mathematical operations

The PLC software supports many mathematical operations, such as addition, multiplication, subtraction and so on, but has limitations for vector or matrix operations. The TIA Portal library lacks built-in vector and matrix operations, whereas Schneider Electric PLCs support vector operations. One of such functions is MUL_ARDINT which multiplies two double-integer arrays. For arrays of the REAL data type, EcoStruxure Control Expert does not provide a built-in instruction equivalent to MUL_ARDINT. To perform multiplication on arrays of the REAL data type one should use a FOR loop to iterate through each element in the arrays, multiplying (or adding) corresponding elements and storing the results in a new array. A code snippet of multiplication of arrays for REAL data types is in Fig. 4. TIA Portal supports different external libraries developed by communities. The Library of general functions (LGF) [25] which was created by Siemens, supports native matrix operations as addition, multiplication and so on, even inverse matrix calculation (LGF_MatrixInverse). This function inverts a square matrix of type ARRAY[*,*] of LREAL using the Shipley-Coleman method, ensuring that the input matrix has equal numbers of rows and columns, while the output matrix retains the same size and array boundaries as the input.

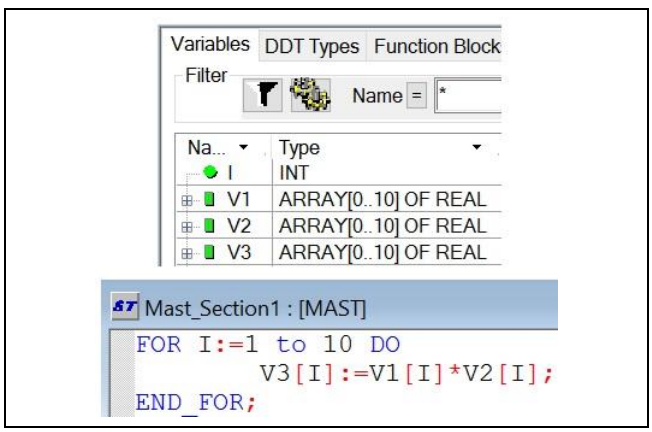


Fig. 4. Example of FOR cycle for multiplication

5 Program organization units (POUs)

A Program Organization Unit in a PLC is a structured block of code used to organize and manage the programming of the PLC. It's also defined by the IEC 61-131 standard. POUs are essential for modular programming, allowing for better organization, readability and maintainability of the control logic. There are typically three main types of POUs [17-19]:

- Function Blocks (FB) – These are reusable code blocks designed to encapsulate specific functionalities allowing them to be called multiple times within a program with different parameters. They have an assigned memory area called instance data block for each call. Among the standard library, timers and counters are the well-known FBs.
- Functions (FC) – These are similar to function blocks but do not retain state information. They are used to perform specific tasks and return a single value. From the standard library mathematical, bit operations and others are examples of functions.
- Programs (OB) – These are the main execution units of the PLC program. An OB defines the sequence of operations and can be called upon under specific conditions such as during startup or based on a specific event.

FC and FB have:

- Inputs – These are values or parameters (variables) provided to the FC or FB that influence its output.
- Outputs – These are the results produced by the FC or FB after processing the input values. They can be used in other parts of the program or can be directly control actions.
- InOuts – These are parameters that serve as both inputs and outputs. They allow data to be passed into the FC or FB, modified and then returned as output. Combining InOuts with structures enables more flexible and dynamic data handling.

In addition, FBs have static variables (or Global Variables in Schneider Electric PLCs). These are states stored in instance DBs. Beside inputs they influence the output values.

To design an AI model, one must choose between the FB approach and the combination of FC with global DBs that contain structured arrays. The FB approach is recommended for encapsulating functionality and maintaining state across multiple calls. Combining FB with structured InOuts with global DBs are also possible. In contrast, using FCs offers the advantage of a flexible interface and customizable internal structure design. Direct implementation within OBs is also possible, allowing for straightforward execution of program without the need for separate FC or FB definitions.

However, for more complex algorithms, structuring the logic using FCs is recommended.

6 Tasks

In PLCs, tasks are important units of execution that determine how and when specific programs are executed. In Siemens PLCs, tasks are closely related to OBs which define the execution behavior of these tasks. In Schneider Electric software, tasks are defined according to IEC 61131 standards. There are three types of tasks in PLCs [17-20]:

- Cyclic Tasks – These tasks are executed continuously and once a task finishes its execution a new execution begins as soon as possible. They are ideal for discrete event systems typically found in industries such as automotive, electrical engineering and manufac-turing, where logical states are processed.
- Periodic Tasks – Unlike cyclic tasks, periodic tasks are executed at fixed time intervals. This makes them suitable for operations that require precise timing, which is crucial for continuous control applications. An example of such applications includes those that utilize PID controllers.
- Event Tasks – These tasks are triggered by specific events, such as a change in input signals or a specific condition being met. Event tasks are particularly useful for responding to alarms or external signals.

Tasks have specific priorities. In Siemens PLCs, one can change these priorities in the OB properties [18]. In Schneider Electric PLCs there are four types of tasks: Mast, Fast, Event and Aux [17]. Mast is cyclic, Fast is periodic, Event is event and Aux is used for background functions that do not require real-time processing. In a multitasking environment, tasks with higher priority interrupt tasks with lower priority ensuring that critical functions are executed promptly. This priori-tization mechanism helps maintain system respon-siveness and reliability, especially in applications requiring real-time control.

Each task has its own watchdog parameter [17-19, 21-24]. A watchdog is a safety mechanism in PLCs that monitors system operations and can trigger a transition from the RUN state of the CPU to the STOP state, causing the program to cease execution and thereby halting system control. The considered AI algorithm should be executed within the watchdog timeout to avoid causing the program execution to stop.

According to the usage of the NN, we can call the program within either a cyclic or a periodic task. In control applications a periodic task is essential. A modern PLC's simplified periodic cycle (task) consists of the following steps [17-21, 23, 24]:

- Reading inputs (RI) – Reading input states using sensors connected to digital and analog input modules, with the process image of inputs being updated.
- Program execution (PE) – Execution of the program in different programming languages.
- Writing outputs (WO) – Writing the process image of output to output modules.

In these modern PLCs the execution order is WO, RI, and PE ensuring a fixed period for reading inputs and updating outputs [17-19].

7 Neural networks

Like in other devices one may face two stages related to NNs. One is the training procedures and the second is the inference. The training can be divided to [26]:

- Online (or incremental) Training is such training, when the model learns continuously by updating its parameters as each new data point is measured. This approach is suited for dynamic environments because it allows the model to adapt in real-time to new patterns and changes (i.e. uncertainties). However, it may struggle to retain long-term knowledge if not managed well. In this case the PLC collects the measured data and prepares the input vector for the NN. One epoch is faster due less input and output data.
- Offline (or batch) Training is such training, when the model is trained on a fixed dataset all at once or in batches. This method is generally more used, efficient for large datasets and allows for extensive hyperparameter tuning. However, it does not adapt to new data until retrained, making it less suited for environments, where data change frequently. In this case one should have the measured data logged in DBs using arrays. These data then are transformed into matrices depending on the task. For example, let us have a single-input and single-output (SISO) system, one may logged the control signal (u) and the systems response (y). One may create a nonlinear neural network based autoregressive exogenous input (NNARX) model in PLC by combining the (u and y) into matrix. One epoch will be slower compared to online training.

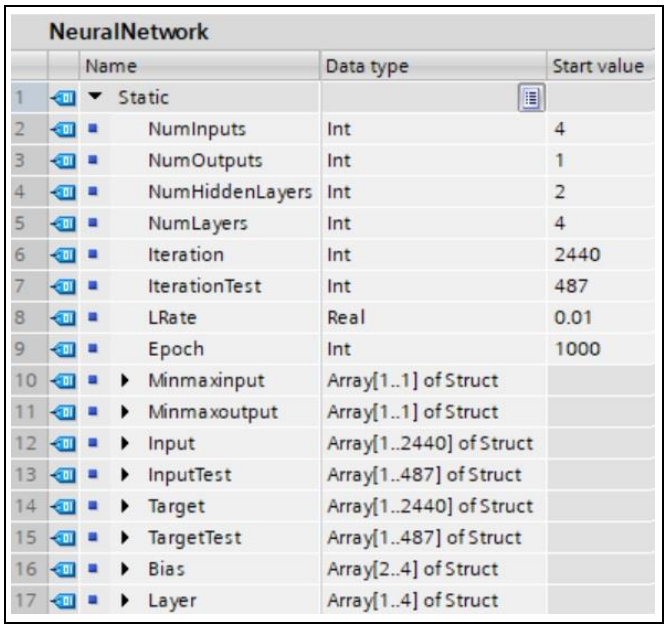
In both cases the designer should take into account the watchdog time. The same applies for the inference. From the inference point of view, it is similar to the online training phase, because one needs to prepare the input vector and calculate the output of the NN. Nowadays PLC CPU are fast enough to deal with the inference of larger NN structures (including deep NNs).

8 Example

Let us consider a multilayer perceptron (MLP) created in Matlab for PLC. One need to define for the inference:

- The number of inputs and outputs (depends on the task)
- The number of hidden layers
- Number of neurons in layers including activation function types
- Hyperparameters (weights and biases) between layers
- Other parameters, e.g., for normalizations and for training:
  - Method type (if more than one is implemented)
  - Parameters of the training method (learning rate, etc.)
  - End conditions of the training (epochs, minimum gradient, etc.)
  - Other parameters, e.g., for normalizations

An example of the DB for the NN inference and training is in Fig. 5. In this example the NN structure and the processed data are stored in one place. The creation of the DB manually (entering the proper settings including hyperparameters) is time consuming and can lead to errors, therefore one may export the NN structure from Matlab to XML format that can be imported into TIA Portal. The trained neural network whether developed using traditional algorithms or Neuro-Evolution based algorithms can be transferred to the PLC [16].



NeuralNetwork

	Name	Data type	Start value
1	Static		
2	NumInputs	Int	4
3	NumOutputs	Int	1
4	NumHiddenLayers	Int	2
5	NumLayers	Int	4
6	Iteration	Int	2440
7	IterationTest	Int	487
8	LRate	Real	0.01
9	Epoch	Int	1000
10	Minmaxinput	Array[1..1] of Struct	
11	Minmaxoutput	Array[1..1] of Struct	
12	Input	Array[1..2440] of Struct	
13	InputTest	Array[1..487] of Struct	
14	Target	Array[1..2440] of Struct	
15	TargetTest	Array[1..487] of Struct	
16	Bias	Array[2..4] of Struct	
17	Layer	Array[1..4] of Struct	

Fig. 5. Examples of NN structure in DB

Part of the XML structure is shown below:

```
DATA_BLOCK "NeuralNetwork"
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1
NON_RETAIN
  VAR
    NumInputs : Int;
    NumOutputs : Int;
    NumHiddenLayers : Int;
    NumLayers : Int;
    Iteration : Int;
    IterationTest : Int;
    LRate : Real;
    Epoch : Int;
    Minmaxinput : Array[1..1] of Struct
      Xmin : Array[1..4] of Struct
        Value : LReal;
      END_STRUCT;
      Xmax : Array[1..4] of Struct
        Value : LReal;
      END_STRUCT;
      Ymin : LReal;
      Ymax : LReal;
    END_STRUCT;
    ...
```

This structure can be integrated as an external source in TIA Portal allowing for the generation of the DB from the source. To handle the learning and training procedure one may implement ST program. An example of a code snippet of the inference is shown in Fig. 6. In this example a FOR loop is used to iterate through all layers to calculate the output(s) of the NN.

The *NeuralNetwork.Iteration* represents the number of samples (column vectors; columns of input matrix). In the first part of the snippet, the input layer is processed. It continues with data normalization using min and max predefined values in DB. A part of a nested loop iterates all layers and finally the output is calculated.

```
// Loop for setting initial neuron values
FOR #iteration := 1 TO "NeuralNetwork".Iteration
    DO
   FOR #input := 1 TO "NeuralNetwork".NumInputs
       DO
      "NeuralNetwork".Layer[1].Neural[#input].
          Value :=
         "NeuralNetwork".Input[#iteration].Neural
             [#input].Value;
   END_FOR;

   FOR #minmax := 1 TO "NeuralNetwork".NumInputs
       DO
      "NeuralNetwork".Layer[1].Neural[#minmax].
          Value :=
         (("NeuralNetwork".Input[#iteration].
             Neural[#minmax].Value -
         "NeuralNetwork".Minmaxinput[1].Xmin[#
             minmax].Value) *
         (("NeuralNetwork".Minmaxinput[1].Ymax -
         "NeuralNetwork".Minmaxinput[1].Ymin) /
         ("NeuralNetwork".Minmaxinput[1].Xmax[#
             minmax].Value -
         "NeuralNetwork".Minmaxinput[1].Xmin[#
             minmax].Value))) +
         "NeuralNetwork".Minmaxinput[1].Ymin;
   END_FOR;

   // Loop for calculating neuron values in the
       neural network
   FOR #layers := 2 TO "NeuralNetwork".NumLayers
       DO
      FOR #neural1 := 1 TO "NeuralNetwork".Layer
          [#layers].Dimensions DO
         FOR #neural := 1 TO "NeuralNetwork".
             Layer[#layers - 1].Dimensions DO
            #tmp := "NeuralNetwork".Layer[#layers
                - 1].Neural[#neural].Value *
                "NeuralNetwork".Layer[#layers].
                    Neural[#neural1].Weight[#
                    neural];
            #sum := #sum + #tmp;
         END_FOR;
         ...
```

Fig. 6. Example of code snippet of the inference

9 Conclusions

Integrating AI methods into industrial control systems presents a promising approach enhancing system quality and efficiency. By leveraging the power of neural networks, industries can achieve higher levels of automation and better decision-making. The deployment of NNs within PLCs offers a unique opportunity to improve the efficiency of control systems enabling real-time data analysis and decision-making. As discussed in the article, it is suitable for structures whose inference and training can occur in real time (within the watchdog time). The implementation of NNs in PLCs requires careful consideration of variables, programming languages, mathematical operations, POUs, tasks and NNs approaches. By following the principles outlined in this paper one can successfully integrate AI methods into their control systems thereby enhancing their capabilities and performance without the need of additional hardware (Edge, PC, etc.).

Acknowledgement

This work was supported by APVV-22-0169 Grant from the Slovak Scientific Grant Agency.

References

[1] L. Körösi, V. Németh, J. Paulusová and Š. Kozák, “RBF neural network for identification and control using PAC”, *International Joint Conference SOCO'13-CISIS'13-ICEUTE'13*, Salamanca, Spain, Berlin : Springer, pp. 329-337, 2013.

[2] L. Körösi, J. Paulusová and Š. Kozák, “PLC online control using orthogonal neural network”, *MENDEL 2013 : 19th International Conference on Soft Computing*, Brno, Czech Republic, pp. 227-232, 2013

[3] L. Körösi, “Neural Network Modeling and Control Using Programmable Logic Controller”, *Posterus*, Vol. 4, No. 12, 2011.

[4] L. Körösi, S. Kajan, J. Paulusová and P. Štefaňák, “Servo System Control Using Gestures”, *2022 Cybernetics & Informatics (K&I) 31st IEEE International Conference*, Visegrád, Hungary, p. 5, 2022.

[5] C. Tian, J. Yi and J. Gao, “An Intelligent Stereo Garage System,”, *2024 43rd Chinese Control Conference (CCC)*, pp. 6439–6444, 2024.

[6] M. Akhmetov, D. Kanymkulov, A. Amirov, A. Askhatova and T. Alizadeh, “Integrated Machine Vision and PLC Commanding for Efficient Bottle Label Detection in Industrial Processes: A Unified Approach for Quality Control”, *2024 10th International Conference on Control, Automation and Robotics (ICCAR)*, pp. 66–70, 2024.

[7] G. A. David, P. M. d. C. Monson, C. Soares, P. d. O. Conceição, P. R. de Aguiar and A. Simeone, “IoT-Driven Deep Learning for Enhanced Industrial Production Forecasting”, *IEEE Internet of Things Journal*, pp. 1–1, 2024.

[8] J. Liguš, T. A. Murajda and S. Filip, “Optimisation of hybrid power system with on site meteo station with integrated prediction methods”, *8th International Hybrid Power Plants & Systems Workshop (HYB2024)*, vol. 2024, pp. 273–279, 2024.

[9] A. Alihodžić, A. Mujezinović and E. Turajlić, “Artificial neural network-based method for overhead lines magnetic flux density estimation”, *Journal of Electrical Engineering*, vol. 75, no. 3, pp. 181–191, 2024.

[10] D. Filimonov, A. Onabek, K. Smolyarchuk and T. Alizadeh, “Integrating Computer Vision in a CODESYS PLC to Enable Intelligent Object Identification,”, *2024 9th International Conference on Mechatronics Engineering (ICOM)*, pp. 65–70, 2024.

[11] V. Kurilová, S. Rajcsányi, Z. Rábeková, J. Pavlovičová, M. Oravec and N. Majtánová, “Detecting glaucoma from fundus images using ensemble learning”, *Journal of Electrical Engineering*, vol. 74, no. 4, pp. 328–335, 2023.

[12] L. Wen, B. Liang, L. Zhang, B. Hao and Z. Yang, "Research on Coal Volume Detection and Energy-Saving Optimization Intelligent Control Method of Belt Conveyor Based on Laser and Binocular Visual Fusion", *IEEE Access*, vol. 12, pp. 75238-75248, 2024.

[13] E. Genc, M. E. Yildirim and Y. B. Salman, “Human activity recognition with fine-tuned cnn-lstm”, *Journal of Electrical Engineering*, vol. 75, no. 1, pp. 8–13, 2024.

[14] W. Cao, “Research on Control Optimization Method of Grinding System Based on PLC and Swarm Intelligence Algorithm”, *2023 7th Asian Conference on Artificial Intelligence Technology (ACAIT)*, pp. 1014–1018, 2023.

[15] C.-T. Lin, M.-F. Han, Y.-Y. Lin, S.-H. Liao and J.-Y. Chang, "Neuro-fuzzy system design using differential evolution with local information", *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*, pp. 1003–1006, 2011.

[16] F. Zúbek, A. Melichar, I. Kenický, L. Körösi and I. Sekaj, "Autonomous Systems Control Design Using Neuro-Evolution", *2022 Cybernetics & Informatics (K&I)*, pp. 1–6, 2022.

[17] Schneider Electric, "Ecostruxure control expert - program languages and structure reference manual", https://download.schneider-electric.com/files?p_Doc_Ref=35006144K01000&p_enDocType=User+guide&p_File_Name=35006144_K01_000_25.pdf, Last accessed: November 2024.

[18] H. Berger, *Automating with simatic: Controllers, software, programming, data*, Publicis, 5th edition, 2013.

[19] H. Berger, *Automating with simatic s7-1500: Configuring, programming and testing with step 7 professional*, Publicis, 2nd edition, 2014.

[20] F. D. Petruzella, *Programmable Logic Controllers*, McGraw-Hill Education, 5th edition, 2016.

[21] J. Hugh, *Automating Manufacturing Systems with PLCs*, Lulu.com, 5th edition, 2008.

[22] B. G. Lipták, *Instrument Engineers' Handbook, Vol. 2: Process Control and Optimization*, CRC Press, 4th edition, 2005.

[23] L. A. Bryan and E. A. Bryan, *Programmable Controllers - Theory and Implementation*, Amer Technical Pub, 2nd edition, 2003.

[24] W. Bolton, *Programmable Logic Controllers*, Newnes, 5th edition, 2009.

[25] Siemens, "Library of general functions (lgf) for simatic step 7 (tia portal) and simatic s7-1200 / s7-1500", https://support.industry.siemens.com/cs/document/109479728/library-of-general-functions-(lgf)-for-simatic-step-7-(tia-portal)-and-simatic-s7-1200-s7-1500?dti=0lc=en-SK, 2024. Last accessed: November 2024.

[26] J. Cai and Z. Deng, "Offline and online modeling of switched reluctance motor based on rbf neural networks", *Journal of Electrical Engineering*, vol. 64, no. 3, pp. 186–190, 2013.

Received 12 November 2024