A Machine Learning Approach and Convolutional Neural Networks for Industry 4.0

MAURO MAZZEI*
* National Research Council, Institute of Systems Analysis and Computer Science, LabGeoInf,
Via dei Taurini, 19, I-00185, Rome, ITALY

Abstract: The aim of this scientific paper is an experimental of artificial intelligence techniques using deep learning and in particular convolutional neural networks (CNN) to optimize industrial processes. An application is presented that can recognize components within an electrical equipment and verify their state. At the same time, the application attempts to identify the coding of industrial components in order to be able to construct an enrichment of the component information. Using an optical character recognition system for detecting and reading the component coding, a search is conducted for the technical specifications of the components. On this aspect, an innovative category prediction system is presented that can recommend the best solution for possible modifications or changes in the event of component malfunctions or failures.

Key-words: Artificial Intelligence, Machine Learning, Deep Learning, CNN.

Received: March 13, 2024. Revised: August 13, 2024. Accepted: September 16, 2024. Published: October 30, 2024.

1. Manuscript

1.1 General Instructions

In recent years, artificial intelligence has become an integral part of our lives and can be found in many sectors. Today, there are many applications that make use of artificial intelligence. A strong use of AI, which combines a kind of intellectual capacity with the ability to interact, is undoubtedly the so-called 'Internet of Things. The Internet of Things refers to several electronic devices, equipped with software and able to connect to the Internet, that have the ability to communicate with each other and exchange data, giving the impression that objects are recognizable and intelligent.

Also, within the Internet of Things, the theme of the smart city is developed, referring to an urban area in which it is possible to make infrastructures and services more efficient in order to build a sustainable city capable of ensuring a high quality of life for its citizens. This is based on connected and integrated technological solutions and systems. These technologies, include information and communication infrastructures (such as 5G), 'big data' analysis, sensors, energy monitoring systems, new materials and solutions for sustainable building, new hybrid and electric vehicles, urban planning models, waste cycle management, and artificial intelligence.

Related to the smart city sector is the smart car branch. In fact, cars are on the rise that already have certain safety functions that use artificial intelligence, such as assisted or autonomous driving systems that detect possible dangerous situations and events, such as automatic emergency braking or sensors for crossing the roadway, connectivity-enabled services such as preventive maintenance based on component monitoring and the use of voice assistants, and smart speakers to interact with one's car using one's voice.

The fourth industrial revolution, in fact, aims precisely at the dissemination of artificial intelligence embedded in objects, which are connected to each other through the Internet and thus become intelligent.

We therefore speak of 'Industrial Internet of Things' (Industrial IoT), the Internet of Things that involves industrial processes in order to improve them in terms of efficiency, productivity, and safety.

2. Machine Learning

Machine learning can be considered a subset of artificial intelligence and is the tool that makes artificial intelligence possible. Machine learning has recently acquired a great importance in the technological context. This interest in the topic is also driven by the fact that we have recently experienced a massive digitalization of everything.

It is the concept of 'data' that machine learning is based on, i.e. the ability of a digital machine to process a large quantity of data and learn rules or functions from it, progressively correcting its algorithm as new data is received. This essentially simulates the human being's ability to learn from experience.

Therefore, machine learning uses a completely different approach to classical programming. In machine learning, the starting point is always input data, but unlike the previous situation, known output data is provided in addition to this. These input-output pairs are then 'studied' by the machine, which this time will autonomously attempt to understand the relationships between these pairs, finally resulting in the algorithm model that best fits the data.

The artificial neuron originates as an attempt to simulate the biological neuron in the brain. Similarly, the artificial neuron, bearing in mind the concept of perceptron, receives inputs that are each multiplied by a coefficient called weight, these weighted inputs are summed up and a function is applied to this summation that will determine the final output.
More precisely in mathematical terms the individual perceptron is defined by the function:

$$f(x) = \begin{cases} 1 & wx + b > 0 \\ 0 & otherwise \end{cases}$$

Where $wx + b$ defines the coordinates of a separating hyperplane that changes position according to the values of w and b. For example, a problem of predicting the output between two possible values, the goal is to calibrate the values of w and b so that the values are correctly separated.
In the context of the perceptron, a problem related to the fact that its output must be either 0 or 1 and not both, so we realize that a small variation of w and b in the vicinity of 0 could cause a drastic jump of the output from 0 to 1 or vice versa.
The most common activation functions are:
Sigmoid function, defined by equation.

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

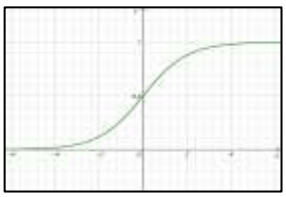

Figure 1: Sigmoid function

Hyperbolic tangent function, defined by equation.

$$\text{Tanh}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

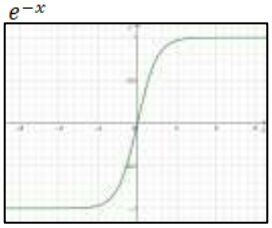

Figure 2: Hyperbolic function

ReLu function, defined by the formula.

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

Figure 3: ReLu function

3. Neural Network Learning

The fundamental characteristic of neural networks is their training capability. The training of the network, known as training, basically takes the form of autonomously adjusting the value of the weights w and bias b. To simplify, a neural network model is represented in figure 4, consisting of two inputs and one output. The two different objects to be classified are represented by two digits: 0 to represent the first object, 1 to represent the second.

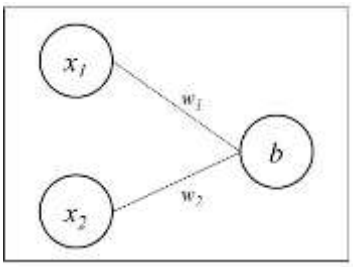

Figure 4: Simple neural network model

In general, by input of x1 and x2 and using the values w1, w2 and b, the summary $\sum$ = x1 w1 + x2 w2 + b is defined, this is given to the activation function (it is assumed that the activation function is the sigmoid function) and a prediction P = sig($\sum$) is returned. P will be a number between 0 and 1 depending on whether the input data is more likely to represent the first or second object.
Initially the neural network does not know the task it has to solve so the weights and bias are random, the network will have to learn the correct values through the successive input of a set of x1 and x2 data from the training set that determine the specific characteristics of the two objects.
For the training phase, the neural network will use the cost function, a function that returns a measure of how far the prediction deviates from the expected result. The closer the cost function is to 0, the more accurate the prediction is.
A cost function that can be used is the quadratic error function, defined as the difference between the prediction P and the target result all squared:

$$c = (P\text{- objective})^2$$

This function then defines a trajectory and if the aim is to minimise this error, then the derivative of this cost function with respect to P calculated at the point determined by the target value comes to the rescue:

$$\frac{dc}{dP} = 2(P - objective)$$

The prediction P should therefore be modified so that, with each new cycle of training data input, the cost function gradually cancels out.
As is well known, the derivative of a trajectory calculated at a point a geometrically turns out to be the tangent line

to the trajectory at the same point a. Therefore the cost function cancels if the derivative is zero, i.e. if P is equal to the objective, and consequently if the slope of the tangent line is zero.
In order to cancel the slope of the tangent line, it is necessary to subtract a fraction of the slope of the line from the actual value of the forecast with the sign. Thus, the next value of P is defined as:

$$P_{succ} = P_{att} - LR\left(\frac{dc}{dP}\right)$$

The fraction mentioned is defined by the LR parameter, called the learning rate, and is a measure of how fast Psucc varies and is therefore crucial in determining the network's learning rate. It is a value between 0 and 1: if it is too low, P will converge to the target value too slowly; if it is too high, Psucc will diverge, never reaching the target value.
As seen above, the prediction P in turn depends, as far as the neural network in figure XXX is concerned, on the values w1, w2 and b; therefore, it is these parameters that must be progressively updated in order to cancel out the cost function.
The process of parameter updating, known as the backpropagation-error algorithm, is divided into two main phases:
Forward propagation, a phase in which the weighted sum of the inputs and the activation function is computed layer by layer, from the beginning to the end, of the neural network, to provide the prediction P. In the final layer, the cost function measures the error made in the outputs from the network compared to the desired outputs.
Backpropagation, a phase in which the error calculated in the output layer is propagated backwards layer by layer and through the chain derivation rule, the gradient of the cost function is computed for each parameter of the network, which will then be updated until it is calibrated with the best possible precision for the resolution of the set task.
The following lines describe the main mathematical steps, that accompany the backpropagation phase.
We will return to the formula defining the cost function:

$$c = -\left(P - objective\right)^2$$

We develop the formula further by making P more explicit:

$$c = (P - objective)^2 = [sig(\sum) - objective]^2$$
$$= [sig(x_1w_1 + x_2w_2 + b) - objective]^2$$

The final aim is to minimise the cost function, so its derivative is calculated:

$$\frac{dc}{dP} = 2[P - objective]$$

This will be used to update the prediction P through the formula:

$$P_{succ} = P_{att} - LR\left(\frac{dc}{dP}\right)$$

Since P, as the neural network under consideration is constituted, depends on the three parameters w1, w2 and b, it is appropriate to specify for each parameter its update function:

$$w_{1,succ} = w_{1,att} - LR\left(\frac{\partial c}{\partial w_1}\right)$$

$$w_{2,succ} = w_{2,att} - LR\left(\frac{\partial c}{\partial w_2}\right)$$

$$b_{succ} = b_{att} - LR\left(\frac{\partial c}{\partial b}\right)$$

The relevant cost functions must then be calculated. We now calculate:

$$\left(\frac{\partial c}{\partial w_1}\right)$$

$$\left(\frac{\partial c}{\partial w_1}\right) = \left(\frac{\partial}{\partial w_1}\right)(\mathrm{P} - object)^2$$

By the chain rule, a rule for calculating the derivative of a function composed of two derivable functions, we obtain:

$$\frac{\partial c}{\partial w_1} = \frac{\partial c}{\partial w_1}(\mathrm{P} - object)^2 = \frac{\partial}{\partial P}(\mathrm{P} - object)^2 . \frac{\partial c}{\partial w_1} P$$
$$= \frac{\partial}{\partial P}(\mathrm{P} - object)^2 . \frac{\partial c}{\partial w_1} sig(x_1w_1 + x_2w_2 + b)$$

The last expression is still a function composed of two derivable functions; therefore, the chain rule applies again. For simplicity, the following substitution is applied:

$$t = (x_1w_1 + x_2w_2 + b)$$

At this point, the result is:

$$\frac{\partial c}{\partial w_1} = \frac{\partial}{\partial P}(P - object)^2 . \frac{\partial}{\partial w_1} sig(t)$$
$$= \frac{\partial}{\partial P}(\mathrm{P} - object)^2 . \frac{\partial}{\partial t} sig(t) . \frac{\partial}{\partial w_1}(x_1w_1 + x_2w_2 + b)$$

The calculation is now carried out:

$$\frac{\partial c}{\partial w_1} = \frac{\partial}{\partial P}(P - object)^2 . \frac{\partial}{\partial t} sig(t) . \frac{\partial}{\partial w_1}(x_1w_1 + x_2w_2 + b)$$

$$\frac{\partial c}{\partial w_1} = 2\,(P - object) . sig(t)[1 - sig(t)] . x_1$$

$$\frac{\partial c}{\partial w_1} = 2[sig(x_1w_1 + x_2w_2 + b) - object] . sig(x_1w_1 + x_2w_2 + b)[1 - sig(x_1w_1 + x_2w_2 + b)] . x_1$$

Similarly, for parameters w2 and b we obtain:

$$\frac{\partial c}{\partial w_2} = 2[sig(x_1w_1 + x_2w_2 + b) - object].sig(x_1w_1 + x_2w_2 + b)[1 - sig(x_1w_1 + x_2w_2 + b)].x_2$$

$$\frac{\partial c}{\partial b} = 2[sig(x_1w_1 + x_2w_2 + b) - object].sig(x_1w_1 + x_2w_2 + b)[1 - sig(x_1w_1 + x_2w_2 + b)]$$

These will be the cost functions for each parameter to be entered into the functions for updating the values, resulting in the updating of a new prediction P.

This procedure is absolutely generalizable also in the case of artificial neural networks consisting of several nodes, the chain rule would always be used but with more derivatives to be computed. By repeating the two steps of the backpropagation-error algorithm over and over again, the parameters will be more and more accurate, allowing for a gradually increasing accuracy. However, after a certain number of iterations on the training data, the generalisation stops improving, the results on the test data stall and then begin to deteriorate. The model is in an overfitting condition, a situation in which the model is starting to learn patterns that are specific to the training data but not relevant to new data. In the underfitting situation, the model is not yet able to distinguish between one class and another, so there is low optimization and generalization.

4. Convolutional neural Netoworks

Convolutional neural networks (or CNNs) are artificial neural networks specialised in solving problems in the domain of image classification and computer vision in general. CNNs can be considered the main example of a deep neural network and an indispensable tool for the use of deep learning: in fact, you can see how computer vision and deep learning are now more interconnected than ever.

Another advantage of CNNs is that they adapt very well to the technique of transfer learning, a tool that makes it possible to reuse previously trained networks on large datasets to solve similar or more specific tasks. In this way, it is not necessary to go and train all the parameters of all layers of the network, but only the last layers, i.e. those that deal with the actual classification.

Convolution is a very efficient solution to the problem of the difficulty of processing images, as before CNNs, time-consuming methods were used to extract the characteristic features of the objects to be identified within the image and the entire image had to be evaluated, while with convolution it is possible to recognise specific patterns delimited in a portion of the image.

Convolutional neural networks introduce new types of layers that have the ability to learn to recognise specific patterns in the image independently of their position, thus being able to find them in other images in different areas, and to decrease the complexity of the network through a smaller number of parameters.

The typical layers that make up a convolutional neural network are explained below.

An example architecture of a CNN is shown in figure 5.

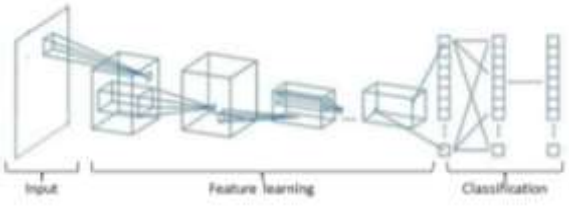

Figure 5: Architecture of CNNs

First of all, as can be seen in figure 5, two main macro-areas can be identified in the CNN. The first part is responsible for feature extraction and is composed of the input layer (the image to be processed) and a more or less deep succession of hidden layers that are not necessarily connected node by node. The last include layers that perform convolution operations, often accompanied by an activation function, including the ReLu function, and layers that perform the downsampling operation called pooling layers. These layers, within the network, can be placed at will in the order and quantity that best suits the task to be solved. The result generated by each layer is called a feature map and this in turn contributes to the input of the next layer.

The second part, on the other side, is to all intents and purposes a classical neural network, essentially composed therefore of Dense layers, also called Fully Connected layers because of their structure of completely connected nodes, and these layers take care of the actual classification starting from the extracted features, which will constitute the input of this neural network. This input layer can be followed by any number of layers and at the end there will be an output layer that will provide the class, to which what is represented in the original image belongs, with the highest probability.

The convolution layer is the fundamental layer of the network and is normally the first layer to appear. Its objective, as mentioned several times throughout this chapter, is to identify features, patterns, which may be curves, angles, lines, edges, textures, etc.

The convolution can be performed several times within the network, each time focusing on identifying a feature of the object in the image. The more times the convolution is performed, the greater the complexity of the feature.

We start from the fact that the input image is treated as a set of pixels defined by numbers organised in a matrix of three dimensions, with each dimension representing one of the three colours of the RGB encoding. Suppose we are dealing with 32 x 32 RGB encoded images: this means that we will have matrices of size 32 x 32 x 3.

At this point, the convolution operation can be performed using a small matrix, called a filter or kernel, normally of size 2x2 or 3x3. This will run through the entire matrix of the input image and at each step will compute the matrix product between the input and filter pixels.

This operation allows the extraction of a specific feature in each area of the image. Since it is repeated several times, an even deeper pattern will be detected in the next step. Once a pattern has been recognised, the convolutional network will be able to recognise it anywhere in the image.
The parameters that define the rules of how the kernel's scrolling over the input matrix and the size of the output (called hyperparameters) are essentially four:
Kernel size: typically the kernel is 2x2, 3x3, or 5x5 pixels. The size of the kernel determines the size of the output;
Step (stride): defines the number of pixels the kernel moves on the input matrix at each iteration. Higher values move the filter with larger jumps and a smaller output matrix is generated;
Depth: corresponds to the number of filters used on the same input matrix to detect different features. If N filters were used, the same procedure had to be repeated N times, obtaining N feature maps, creating an output N deep;
Zero-Padding: it has been said that kernel size and pitch affect the width and height of the output volume: the kernel goes through the input matrix anchoring itself to the input cell with its centre; therefore, for example, as far as the first cell at the top left of the input is concerned, the filter matrix would be outside the input.
By configuring these hyperparameters, an output volume is obtained, in which the height and width are calculated from the relations:

$$H_{out} = \frac{H_{in} - K + 2P}{S} + 1$$

$$W_{out} = \frac{W_{in} - K + 2P}{S} + 1$$

Where Hout and Wout define the output height and width, Hin and Win the input height and width, K the kernel size, P the padding thickness, S the stride value. The depth of the output volume (depth) is defined by the number of filters used NF.
The values K, P and S must be chosen such that the relations return an integer value. ReLu layer
The ReLu layer is a layer that is usually placed after the convolutional layer and is responsible for applying the ReLu activation function, thus introducing a non-linearity factor to the network.
The layer receives as input the numerical values that are produced by the output volume generated by the convolution operation and applies the following function to each one:

$$f(x) = \begin{cases} x \ per \ x \ \geq 0 \\ 0 \ per \ x \ < 0 \end{cases}$$

In this mode, a matrix of the same size as the previous one is generated, all negative values are cancelled out while the positive ones remain unchanged.
Introducing these ReLu levels, convolutional neural networks perform much better: the network is able to train much faster without significantly affecting the accuracy of the results. Faster learning has a great influence on the performance of large models trained on large datasets.
The pooling layer is a layer often alternated in models with the convolution and ReLu layers and its objective is to progressively reduce the spatial dimension of the matrix. This operation is also known as downsampling, indicating an under-sampling, i.e. a reduction in the number of parameters to be processed and thus in processing costs.
What is obtained from the succession of convolution operations and the application of the ReLu activation function is a matrix or matrices of values in which no negative numbers are present and which faithfully summarise the characteristics of the image. To make the model even more precise, the pooling method can be used.
The pooling operation takes place in a similar way to the convolution operation, i.e. through a filter that sweeps the entire input matrix, but the operation that is performed between the input matrix and the pooling filter is to aggregate the values within the filter by bringing out the largest value among them. In this way, the next smaller layer will include only the largest values among those processed by the filter. The pooling operation is applied separately to each feature map.
To be precise, this operation just described is the max-pooling operation. As an alternative to the max-pooling operation, the average pooling operation is sometimes used, which instead of considering the maximum value of the portion of the matrix considered, considers its average value. In the most common cases, however, the max-pooling operation is preferred to the average pooling operation.
A lot of information is lost in the pooling layer, but there are also a number of advantages for CNN. Pooling layers help to reduce complexity, improve efficiency, and limit the risk of overfitting.
The layer fully connected is, as mentioned before, a traditional multilayer neural network, in which therefore each neuron of the previous layer is connected to each neuron of the following layer, which uses, in more detail, an activation function called "Softmax" in the output layer.
The output of the convolutional, ReLu and pool layers represents the high-level features of the input image. The purpose of the fully connected layer is to use these features to classify the input image into various classes according to the training dataset.
The output of the convolution and pooling layers mentioned earlier are 3D volumes, a fully connected layer on the other hand involves a vector of numbers. So first the output of the final layer is flattened into a vector and this becomes the input for the fully connected layer.
At the output layer of the fully connected layer, the Softmax activation function is finally executed, which takes a vector of arbitrary scores with real values and "flattens" it to a vector of values between zero and one

that sums to one: thus, the sum of the output probabilities of the fully connected layer is 1.

In the context of convolutional neural networks, to try to counteract the problem of overfitting, a phenomenon described in the previous chapter, the most popular technique is to insert dropout layers between the hidden layers of the network. Although the dropout method is effective in preventing overfitting, it is not always sufficient. This is because overfitting occurs when the network trains on little data. The only way to avoid the problem of overfitting is to have an infinite amount of training data, in which case the network would be training on every possible instance of the phenomenon.

Therefore, another method to reduce overfitting is to use more training data, but in an image classification task obtaining new data, i.e. new images, is not easy.

This is precisely what the technique of data augmentation deals with: artificially increasing the size of one's dataset. The technique involves, starting from an image, the reproduction of further images through random transformation of existing ones, which are subsequently supplied to the model as input. The aim is not to make the network process the same image twice, but to make it analyze images in different positions and rotations to improve its generalization capability.

The main transformations include operations such as rotation, translation, reflection, zooming, image flipping, but also cropping, adding noise, deformation, blurring.

5. Case Study

The work performed focused on computer vision technologies from which knowledge and best practices were extracted and transferred to the application domain.

In order to realise the goal of machine vision-based electrical component detection, the idea was to identify electrical components by category by creating an image database as the foundation of deep learning-based object detection. Collecting real photos containing most of the electronic components to build a training dataset is difficult. Moreover, even experts are sometimes unable to identify all devices by categories by checking the labelling. For these reasons, we thought of using virtual components extracted from the network in order to have additional information to better classify the components.

Electrical components come in a wide variety of suppliers, shapes and sizes, so no universal dataset is currently available. The labelling work will have to be done through a GUI with even non-expert operators, so it takes time and effort to complete the job. For these reasons, it was decided to create management software to prepare the environment for the collection of the valuable data to be classified. The database of classified images was used for training by combining real and virtual data.

The label information is clearly visible on all devices and the text content is very complete, especially in the accompanying information extracted from the network, for our detection objective (identification of electrical components) the screen-printed identification number is sufficient, other information is sometimes redundant but useful for links to datasheets.

Virtual images are visually almost identical to the real thing. The synthetic data of the virtual image carries the part identifier the name of the parts and the category; therefore, for the labelling phase of deep learning it is easier to classify them, even a user with no electrical knowledge can directly name the label facilitated by the searches and classification done on the network. Combining virtual and real data to form a training dataset can reduce the cost of data collection and improve the generalisation of the learning model. For the hierarchies of categories associated with the electrical component identifier, it is also thought to evaluate similarity reasoning techniques to be used in support of classification algorithms to make category prediction, it is believed that Lin's measure of similarity, previously introduced by Resnik, between concepts based on the notion of information content exploiting a taxonomy, or domain ontology, can be used. A domain ontology, and in particular a taxonomy of concepts organised using the ISA hierarchy, and a lexical database are considered.

In the labelling phase, the bounding box is used to describe the position of the target in the object detection. The bounding box is a polygon that can be determined by identifying the x and y coordinates of the upper left corner and the height and width (bx, by, bh, bw). The input image is divided into a grid of S × S cells for each object in the image one grid cell is 'responsible' for its prediction, each grid cell predicts bounding and category probabilities.

In particular for both performance and network suitability in the specific implementation goal, we evaluated the YOLO V3 network which uses 53-layer Darknet-53 as a feature extractor. The two most important parts of Darknet-53 are Convolutional and ResNet. The 1 × 1 convolutional can compress the number of channels in the feature map to reduce the model calculation and parameters. Multiple 3 × 3 convolution results in non-linear for a large filter convolution layer, making the decision function more resolvable. ResNet can make the network deeper, faster, easier to optimise, with fewer parameters and less complexity than other models. Therefore, it can solve the deep network problem in degradation and learning difficulty. Darknet-53 performs a total of five dimensionality reductions on operations. The number of rows and columns belong to the feature matrix of each dimension. The output for the reduction becomes half while the depth doubles from the previous one. The figure below shows the corresponding feature output of the different dimension reduction modules of the entire Darknet-53.

By means of the Darknet-53 feature output diagram, we can see that each pixel on the feature map of different layers represents different dimensions of the original image. For example, each pixel in the 104 × 104 × 128 image represents the 4 × 4 size of the original 416 × 416 image. At the same time, each pixel in the 13 × 13 × 1023

image represents the 32 × 32 size of the original 416 × 416 image. For this reason, the network chose 52 × 52 × 256, 26 × 26 × 512 and 13 × 13 × 1024, three layers of feature output for upsampling and feature fusion to detect large, medium and small objects of different sizes. This solution fits very well in our implementation goal as it needs to detect components belonging to a large frame, while the size of the components is small. In the preprocessing phase, each image used for training has different resolutions and different sizes due to the different acquisition method. Therefore, before training, the network resizes the image size to 416 × 416. This facilitates the label content in the original image.

For the analysis of the entire assembly or electrical panel to be monitored, we thought of using change detection algorithms that are best suited to the verification of generalised contexts without excluding the components, but which return a change of state, in particular we are evaluating two methods: deterministic and probabilistic, both of which refer to a definition of image as a set of random variables linked to regions of the plane, in the case of 2D images.

The diagram below represents the proposed image recognition pipeline, a training dataset is proposed divided into two categories for the realisation of the model learning dataset:

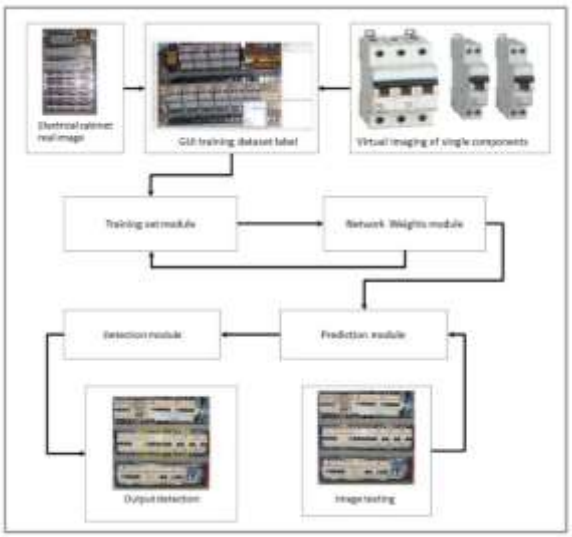

Figure 6: Model of methodology used

The first category is defined by real images taken in the field in environments with different illumination and different scenarios. The filming tools can be of different types, and the images should contain differences in size and resolution

The second category is defined by virtual images taken from the online repositories of distributors selected by the Lucana Sistemi partner (Bticino, Siemens, ABB, Gewiss, Scheneider Electric, Lovato).

The two categories are used to train the deep learning network, the error between the predicted value and the actual value is calculated by the loss function (MSE, MAE, Huber Loss), using the backpropagation of the error in the neural network and constantly adjusting the weight of each convolutional layer of the network to complete the training of the model. With this process schematised in the figure representing the algorithm's pipeline, the loss function determines the direction in which the model is trained.

In the context of evaluating similarity reasoning techniques to be used in support of classification algorithms for category prediction, it is believed that Lin's measure of similarity, previously introduced by Resnik, between concepts based on the notion of information content that exploits a taxonomy, or domain ontology [Resnik, Lin], can be used. This is illustrated below.

Let us consider a domain ontology, and in particular a taxonomy of concepts organised using the ISA hierarchy, and a lexical database such as, for example, WordNet [Fellbaum]. In addition to nouns, WordNet contains verbs, adjectives and adverbs, each associated with its own natural language definition and frequency. Frequencies are estimated using huge text collections, such as the Brown Corpus of American English. In WordNet, concepts are organised not only through ISA relations, but also through Part-of relations, SynSet, etc.. Lin's similarity measure, in particular, exploits the ISA relation. It is assumed that given a domain ontology of concepts organised via the ISA relation, each concept c is associated with a probability p(c) defined as:

$$p(c) = freq(c)/M$$

where freq(c) is the frequency of c for example in the Brown Corpus of American English and M is the total number of concepts in the corpus. Such an ontology, once a probability has been associated with each concept, is called a weighted ontology.

After the probabilities have been associated with the concepts, the initial assumption of the approach is that the information content of a concept c, IC(c), is defined as:

$$IC(c) = - \log p (c)$$

i.e. as the probability of a concept increases, its information content decreases, so the more abstract a concept is, the lower its information content (the root of the taxonomy is the most general concept that has probability 1). According to this approach, the similarity of two hierarchically organised concepts is given by the maximum information content shared by the concepts, i.e. the more information content the two concepts share, the more similar they are. Given two concepts, the maximum information content shared by them in the taxonomy is the least upper bound (lub), i.e. the most specific superconcept of both concepts in the taxonomy. Under these assumptions, the similarity between two concepts c1 and c2, Sim(c1,c2), is defined by the information content of the lub of the concepts divided by the sum of the information contents of the concepts we wish to compare:

$$Sim(c1,c2) = 2 \log p(lub(c1,c2))/(\log p(c1) + \log p(c2))$$

A number of natural language processing (NLP) techniques such as word counting, latent semantic analysis (LSA), word vector representation and character

sequence analysis can be used to calculate similarity between lemmas.
Once word vectors have been created, cosine distance can be used to calculate the similarity between lemmas. Cosine distance is a measure of the similarity between two vectors in a multidimensional space and is defined as the cosine of the angle between the two vectors. The more similar the two vectors are, the closer the angle between them approaches zero and the value of the cosine distance approaches 1.
It should be noted that using machine learning models to create the vector representation of words may require adequate data preparation and training of the model on a large text dataset.

6. Conclusion and future works

The object detection and image classification model described has proven to have good efficiency in that satisfactory results were always provided. Convolutional neural networks have therefore proved the quality of their technique, confirming the ability of machine learning, and in particular deep learning, to be a technology capable of supporting the human decision-making level. The computer vision model realised in this work could be adaptable in other areas related to industry and in the development of new implementations and functionalities. In particular, a natural evolution of this application could be the real time monitoring of the status of switches, so that it could be used in the field of security, where, for example, it is checked that the installations are all disconnected before carrying out maintenance operations, and also for the maintenance of the installations themselves.
As a future development, this computer vision technology could be implemented in industry for the monitoring of component production in high-speed continuous cycles for the detection of component anomalies. In addition, other possibilities open up for increasing knowledge in favour of industrial processes by bringing together other methodologies also tested in this work, such as category prediction and taxonomic classification of lemmas, there is no doubt that integrating the two solutions would significantly increase the knowledge base.

References

[1] O. J. Bedrij, "Carry-select adder," IRE Trans. Electron. *Comput*, pp. 340–344, 1962
[2] B. Ramkumar, H.M. Kittur, and P. M. Kannan, "ASIC implementation of modified faster carry save adder," *Eur. J. Sci. Res.*, vol. 42, no. 1, pp.53–58, 2010.
[3] T. Y. Ceiang and M. J. Hsiao, "Carry-select adder using single ripple carry adder," *Electron. Lett.*, vol. 34, no. 22, pp. 2101–2103, Oct. 1998.
[4] J. M. Rabaey, *Digtal Integrated Circuits— A Design Perspective* Upper Saddle River, NJ: Prentice-Hall, 2001
[5] M. Somalvico, L'intelligenza artificiale, Rusconi Editore, 1987
[6] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, in Bulletin of Mathematical Biophysics, vol. 5, 1943, pp. 115-133
[7] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon, A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, AI Magazine, vol. 27, no. 4, 31 agosto 1955, p. 12
[8] F. Rosenblatt, The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, in Psychological Review, vol. 65, no. 6, Cornell Aeronautical Laboratory, 1958, pp. 386–408
[9] G. E. Hinton, S. Osindero, Y. Teh, A fast learning algorithm for deep belief nets, in Neural Computation, vol. 18, no. 7, 2006, pp. 1527-1554
[10] A. Gulli, A. Kapoor, S. Pal, Deep Learning with TensorFlow 2 and Keras, Packt, 2019
[11] R. Talarico, Introduzione Alle Reti Neurali 02: La Funzione di Costo, 21/05/2018
[12] F. Chollet, Deep Learning with Python, Manning, 2018
[13] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in neural information processing systems, pp. 1097– 1105, 2012
[14] A. Kaehler, G. Bradski, Learning OpenCV 3: Computer Vision in C++ with the
[15] OpenCV Library, O'Reilly Media, 2016
[16] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in CVPR, 2016, pp. 779-788
[17] T. Carneiro, R.V. Medeiros da Nòbrega, T. Nepomuceno, G.B. Bian, V.H.C. Albuquerque, P. Filho, Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications, in IEEE Access, 2018, pp. 61677-61685
[18] M. Mazzei, The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 2022-10-14 | Journal article DOI: 10.5194/isprs-archives-XLVIII-4-W5-2022-105-2022
[19] Mazzei, M., Advances in Intelligent Systems and Computing 2021 | Book DOI: 10.1007/978-3-030-55187-2_21 EID: 2-s2.0-85090096064 Part of ISSN: 21945365 21945357
[20] Mazzei, M., Advances in Intelligent Systems and Computing 2021 | Book DOI: 10.1007/978-3-030-73103-8_58 EID: 2-s2.0-85105926014 Part of ISSN: 21945365 21945357
[21] Massimo De Maria; Lorenza Fiumi; Mauro Mazzei; Bik Oleg V., Heritage 2021-07-28 | Journal article DOI: 10.3390/heritage4030079
[22] Mazzei, M., Advances in Intelligent Systems and Computing 2020 | Book DOI: 10.1007/978-3-030-

52246-9_41 EID: 2-s2.0-85088511397 Part of ISSN: 21945365 21945357

[23] Mazzei, M., Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 2019 | Book DOI: 10.1007/978-3-030-24302-9_31 EID: 2-s2.0-85069196224 Part of ISSN: 16113349 03029743

[24] C. Capodiferro and M. Mazzei, "Applications of deep learning and artificial intelligence methods to smart edge devices and stereo cameras," *2023 8th International Conference on Smart and Sustainable Technologies (SpliTech)*, Split/Bol, Croatia, 2023, pp. 1-5, doi: 10.23919/SpliTech58164.2023.10193298.

[25] U. Karn, An Intuitive Explanation of Convolutional Neural Networks, 11 agosto 2016, https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/, consultato in data 30/12/2021

[26] https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60, consultato in data 30/12/2021

[27] https://opencv.org/about/, consultato in data 23/11/2021

[28] https://en.wikipedia.org/wiki/OpenCV, consultato in data 23/11/2021