Vision Inspection with Neural Networks

Arnaud Nguembang Fadja[1], Evelina Lamma[1], and Fabrizio Riguzzi[2]

[1] Dipartimento di Ingegneria – University of Ferrara
[2] Dipartimento di Matematica e Informatica – University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy
[arnaud.nguembafadja, evelina.lamma, fabrizio.riguzzi]@unife.it

Abstract. This work describes a system for extracting and classifying defects inside bottles for cosmetic and pharmaceutical use. The system integrates various defects identification and automatic classification algorithms based on neural networks (NN). The aim is to be able to identify defective bottles at the end of production chain. In a set of 60 bottles, 3600 images were taken, 60 for each. We extracted 4161 defects of which, 70% was used for training and 30% for testing the neural network. We considered five defect classes (rubber, aluminum, glass, hair and tissue) with more than 90% accuracy on the test set.

Keywords: Vision Inspection, Computer Vision, Neural Networks

1 Introduction

Computer vision (CV) [16] aims at understanding information in images, for example, extracting patterns. In manufacturing, CV plays a major role in various applications of measurement, location, identification and inspection (or control) thanks to increasingly sophisticated automatic classification and real-time vision algorithms. The industrial problems faced by CV are heterogeneous. In *measurement* applications, the aim is to measure *physical features* of the objects. Example of features are diameter, area, volume, height... In *detection* applications, the purpose is to locate the object in an area by reporting its position and its orientation. center of gravity or corners and then send these information to a robot for picking up the object. In *identification* applications, the vision system reads various *codes* and alphanumeric characters, for example barcodes, machine plates and matrix codes. In *inspection* or control applications, the aim is to validate certain features, for example the *presence* or *absence* of a correct label on a bottle or the *classification* of defects on the surface or inside a product.

Automated visual inspection of industrial products for quality control plays an important role in production processes. Manufacturing firms try to automate as much as possible the process of production in order to decrease the production cost. However, at the end of the production chain, products can be affected by many defects due to:

1. Degradation of machines used in the production chain,

2. Degradation of machines not directly involved in the production chain,
3. Contamination from the environment.

Therefore, at the end of the production chain, products have to be control in order to guarantee a good quality. In most cases, this quality inspection is carried out by humans by visual inspection. In the cosmetic and pharmaceutic industry, each product is inspected by many employees and the product contains a certain defect if the majority detects it. However, the reliability of manual inspection is limited because of fatigue and inattentiveness. Therefore, firms are working hard towards the automation of the visual inspection process in order to obtain high quality products with high-speed production. In this paper we present a vision system for inspecting products for pharmaceutical and cosmetic use. The system identifies and classifies defects inside bottles. These defects can be of various types: rubber, aluminum, particles or hair and tissue fibers. Figure 1 shows examples of defects in water bottles.

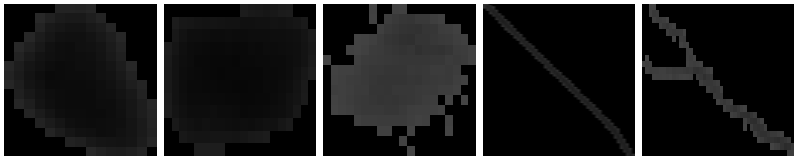

Fig. 1. From left to right: rubber, aluminum, glass particle, hair and tissue fiber.

The paper is organized as follows: Section 2 introduces Computer Vision, Section 3 describes Neural Networks, Section 4 discusses related work, Section 5 presents the experiments performed and Section 6 concludes the paper.

2 Computer Vision

Computer Vision is the process that aims at creating an approximate model of the real world (in 3D) from bidimensional (2D) images. CV can replace or complement manual inspections and measurements with digital cameras and image processing. The technology is used in a variety of different industries to automate inspection, increase production speed and yield, and improve product quality. The main purpose of CV is to reproduce human vision in analyzing and understanding the content of the acquired image. Information is understood in this case as something that permits a classification, for example recognizing an object in an image.

Unlike CV, Images Processing (IP) receives as input an image and outputs a processed image, for example with a modified contrast or brightness, and rather than understanding the image content.

CV uses some techniques from IP, such as filters, before using proper CV techniques. The different steps of CV can be summarized as follow:

1. *Acquiring the image*
   In CV systems, *acquiring* or *grabbing* images of a scene is the first and one of the most important step. Good *digital cameras*, *lens* and *lightning* are needed to obtain high quality digital images. The type (black and white, gray-level or color) and the quality of the image are important decisions to make during acquisition. While black and white images take less space and processing time, color images take large space and processing. Gray-level images often provide a fair compromise. The choice of type of image is based on various criteria like performance and information content to be extracted.
2. *Processing the images (IP)*
   The second step in a CV system is to improve the quality of images through IP. Usually, we are interested only in specific regions of the image called Regions of interest (ROI). A ROI contains the information to analyse. For example, if we aim at identifying particles in bottles, after acquiring the whole image of the bottle, the ROI is the part of the image that contains particles. We then use image processing in order to increase the quality. In this step we can change the contrast, the brightness or the rotation.
3. *Extracting information from the image*
   In this step the system analyses the ROI. For example, if the ROI contains different particles, each can be extracted using a segmentation process. *Segmentation* is performed by selecting pixels of relevant object (the foreground). In particle identification, for example, particles can be highlighted in the ROI by extracting brighter pixels. After segmentation, the system extracts features of the ROI. These features, or vector of feature, depend on the application, and are then used for classification or identification purpose.
4. *Taking action*
   In the last step we use different techniques, such as deep learning 3, for classifying or clustering the ROI. Decision can then be taken, for example deciding whether region contains a cat or a dog.

3 Neural Networks

Machine learning [12] is the ability of extracting knowledge from data. For years, constructing a pattern-recognition [4] or a machine learning system was a difficult and complicated task. Researchers had to design feature extractors that transformed the raw data in an internal representation, a feature vector, from which a learning subsystem, for example a classifier, could be applied. Rather than building two subsystems, raw data can be fed to a single system that is able to discover the representation needed for detection or classification. Deep learning [10] uses this techniques and feeds raw data to a sequence of layers. Each layer is composed by different non linear elements. Elements in a single layer provide the input of the next layer and learning consists of training from raw data.

An Artificial Neural Network (ANN) [17] is an information processing paradigm that is inspired by the way a biological nervous systems, a brain, processes information. It is composed of a large number of highly interconnected processing

elements (neurons) working together to solve specific problems. It can be used for classification or regression. In classification purposes, the network is organized as follow:

1. The input layer is composed of a non processing neuron for each input feature.
2. One or more hidden layers. Each layer has one or more processing neurons.
3. The output layer is composed of different processing neurons one for each class.
4. The output of each neuron in a layer is connected to the input of all the neurons in the next layer (in fully connected networks). Each connection is associated to a weight.

In classification, the aim of training is to modify the weights in order to correctly improve the classification of the training set. The intuition is that if the network can correctly classify all the examples of the training set, or most of them, it will be able to classify well unseen data. This means that it has acquired *generalisation* capacity. Depending on the information flow among the different layers, we have two topologies of ANN.

Feedforward ANN (FANN) [2]: this is the simplest type of ANN in which information moves in only one direction—forward: from the input nodes, data goes through the hidden nodes (if any) and to the output nodes, see Figure 2.

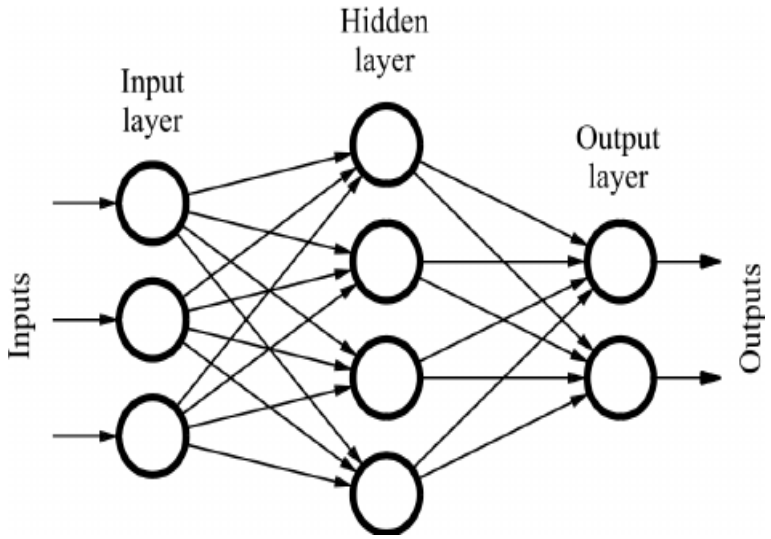

Fig. 2. Feedforward neural network

Recurrent ANN (RANN) [11]: while a feedforward network propagates data from input to output, RANNs also propagate data from later processing stages to earlier stages.

The most used FANN is the MultiLayer Perceptron (MLP) [6]. The nodes of MLP neural networks are perceptrons [15]. A perceptron is a single neuron, having n inputs and one output. It is composed of:

1. N inputs, $x_1 ... x_n$.
2. Weights for each input, $w_1 ... w_n$.
3. A dummy input x_0 (constant and equal to 1) and associated weight $w_0 = -\theta$.
4. Weighted sum of inputs, $net = \sum_{i=0}^{n} w_i x_i$
5. The transfer (or activation) function computes the value of the output signal based on the previous value.

$$y = T(net)$$

$$T(net) = \begin{cases} -1, & \text{if } net < 0, \\ +1, & \text{if } net \geq 1. \end{cases}$$

In order to introduce non linearity in our system, we consider the following activation function (called sigmoid function):

$$T(net) = \sigma(net) = \frac{1}{1 + e^{-net}} \tag{1}$$

To train a MLP, data is usually divided into two sets: the *training* and the *test* set. The training set is used to train the system and the test set is used to test its ability to generalize on unseen data. Depending on whether the data is labeled or not, we can distinguish *supervised learning*, in which input objects (typically vectors) are labeled with a desired output value and *unsupervised learning*, that discovers hidden structures from unlabeled data.

MLP can be trained using techniques such as *backpropagation* [14]. To update the weights, errors are back propagated from the output to the input layers in order to minimize the error of the output. Back-propagation applies gradient descent [1] to reach a local minimum in the space of parameters. Various techniques such as learning rate [7] and momentum [13] can be used to avoid bad local minima.

4 Related Work

Several system are related to ours. ImageNet [8] uses Convolution Neural Network (CNN) [3] for training and classification. CNN are neural networks designed to process data that comes in the form of arrays such as 1D for signals, 2D for images and 3D for videos. There are two main types of layers: *convolutional layers* and *fully connected layers*. Convolutional layers extract conjunctions of features from the previous layer and the fully connected layers classify them. In [8] a deep convolutional neural network is used to classify 1.3 million high-resolution images in the LSVRC-2010 ImageNet training set into 1000 different classes.

[3] `http://cs231n.github.io/convolutional-networks/`

In [9] authors apply CNN to the MNIST dataset, a database of handwritten digits with a training set of 60,000 examples, and a test set of 10,000 examples.

Recurrent neural network [5] can also be applied to image classification.

5 Experiments

We used the Halcon 12 [4] for acquiring, processing and identifying ROIs. It also provides procedures and functions for training and testing MLPs with *one hidden* layer. The system is composed of different subsystems described in the following subsections:

5.1 Acquisition and Defects Identification

This subsystem extracts defects in images of the bottles. Images are acquired with a mechanical devices that rotates the bottle around its vertical axis in front of a digital camera. When the rotation stops, the liquid inside the bottle continues to rotate and the camera acquires a set of 20 successive images, see Figure 3. This procedure is repeated for the same bottle 3 times, acquiring a total of 60 images per bottle. We acquired images from a set of 60 bottles with the following distribution of particles: 10 bottles contain rubber particles, 10 aluminum particles, 20 glass particles, 10 hair fibers and 10 tissue fibers. Overall we collected 3600 images, divided into 180 sets of 20 images.

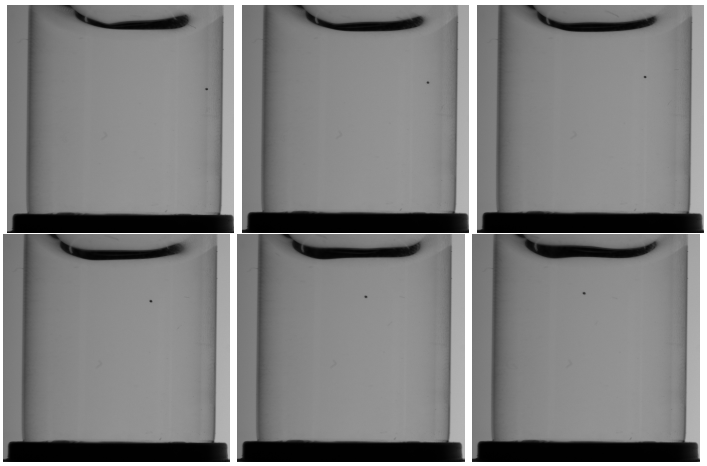

Fig. 3. Defect moving in the bottle

In each image in 3 we are interested in the part showing the liquid: the region under the meniscus. This region is called Region of Interest (ROI) 4. We extracted 3600 ROI overall.

[4] `http://www.mvtec.com/products/halcon/product-information/version12/`

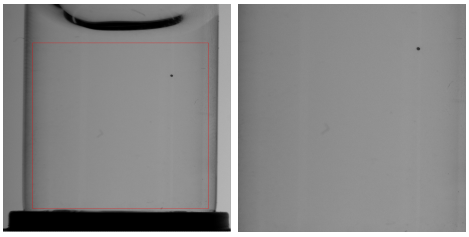

Fig. 4. Region of Interest.

In order to identify defects, ROIs in the same set of 20 ROIs are compared in pairs, one ROI with the next, and differences, in term of pixels, represent defects that are moving in the bottle. The minimum rectangle that contains each defect is extracted and then saved as a small image see Figure 1. From the 3600 ROI we have extracted further 4161 defect images distributed as shown in Table 1.

Defect type	*Number of images*
Rubber	951
Aluminum	556
Glass	1305
Hair	384
Tissue	965

Table 1. Distribution of extracted defects

5.2 Feature Extraction

This subsystem extracts the features of each defect image. We considered the 121 default features provided by Halcon. These features are divided into three groups: region, gray-level and contour features. 63 *region features* describe characteristics of the region such as *area* (the number of pixels), *circularity* (how much the region looks like circle) and the *position* of the region in the image. 19 *gray-level features* describe the properties of the region such as the *min* or the *max* value of pixels and the *standard deviation*. 39 *contour features* describes contour such as *maximum diameter*, *orientation*, *compactness*, and *convexity*.

From this set of feature, we removed 8 features that we deemed not useful, for example *position* (row and column) and *orientation* of the region and its contour in the small rectangle image. After the extraction of the features, to speed up the training as well as the classification we used Principal Component Analysis (PCA) [3] for preprocessing of the feature vectors.

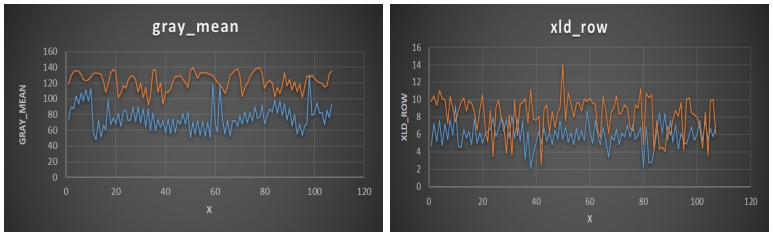

Fig. 5. gray_mean and xld_row features for black and light particles.

5.3 Classification

We built a MLP with the parameters in table 2. The parameter *NumInput* specifies the dimensionality of the feature vectors. *NumHidden* defines the number of units of the hidden layer of the MLP. Since it significantly influences the result of the classification, it should be adjusted very carefully. Its value should be between *NumInput* and *NumOutput*. Smaller values lead to a less complex separating hyperplane and large value run the risk of overfitting. *NumOutput* specifies the number of classes. The parameter *OutputFunction* determines the function used by the output units. In almost all classification applications, *OutputFunction* should be set to *softmax*. *Preprocessing* defines the type of preprocessing applied to the feature vectors for training as well as for testing. The parameter *NumComponents* defines the number of components to which the feature vector is reduced if a preprocessing is applied. In particular, NumComponents has to be adjusted only if a PCA is performed. *RandSeed* is the seed for the random number generator.

Parameters	*Value*
NumberInput	113
NumHidden	100
NumOutput	2,3,5
OutputFunction	softmax
Preprocessing	principal_components
NumComponent	100
RandSeend	100

Table 2. MLP parameters

Table 3 shows the training and classification parameters. Training parameters *MaxIterations*, *WeightTolerance*, and *ErrorTolerance* control the nonlinear optimization algorithm. *MaxIterations* specifies the number of iterations of the optimization algorithm. The optimization terminates if the weight change is smaller than *WeightTolerance* and the change of the error is smaller than *ErrorTolerance*. In any case, the optimization terminates after at most *MaxIterations* iterations.

Parameters	*Value*
MaxIterations	400
WeightTolerance	0.0001
ErrorTolerance	0.01

Table 3. Training parameters

According to the number of classes (2,3,5), we implemented 3 architectures: one for experiment.

5.4 Results

We divided the data into two sets, the training set with 70% of the examples (2909) and the testing, 30% of the examples (1252).

In the first experiment we considered two classes: *particles* (rubber, aluminum, glass) and fiber (hair, tissue). We obtained *99%* precision for particles and *93%* precision for fibers for a total accuracy of *97%*. The lower precision on fibers is due to the fact that small fibers are sometimes classified as particles. In the second experiment we considered three classes: black particles (rubber, aluminum), light particles (glass) and fibers (hair, tissue) and we obtained *95%* precision for black particles, *91%* precision for light particle and *94%* precision for fibers, for a total accuracy of 93%. We then considered five classes (rubber, aluminum, glass, hair, tissue) in the last experiment. We obtained *89%* precision for rubber, *86%* precision for aluminum, *91%* precision for glass, *85%* precision for hair and *97%* precision for tissue for a total of accuracy of *91%*.

6 Conclusions

We have presented a system for inspecting defects in products for pharmaceutical and cosmetic use. The algorithm identifies and classifies defects using a MLP architecture with *one hidden* layer. Experiments show that the MLP can achieve good results. We obtained an accuracy of more than 90% on five different classes (rubber, aluminum, glass, hair, fiber). In the future we plan to add other defect classes, such as *plastic particles* or *bubbles*, and perform experiments with *deep* architectures such as convolutional neural networks.

References

1. Baldi, P.: Gradient descent learning algorithm overview: A general dynamical systems perspective. IEEE Transactions on Neural Networks 6(1), 182–195 (1995)
2. Bebis, G., Georgiopoulos, M.: Feed-forward neural networks. IEEE Potentials 13(4), 27–31 (1994)
3. Everitt, B.S., Dunn, G.: Principal components analysis. Applied Multivariate Data Analysis, Second Edition pp. 48–73 (1993)

4. Flusser, J., Suk, T.: Pattern recognition by affine moment invariants. Pattern Recognition 26(1), 167–174 (1993), https://doi.org/10.1016/0031-3203(93)90098-H
5. Grossberg, S.: Recurrent neural networks. Scholarpedia 8(2), 1888 (2013)
6. Haykin, S., Network, N.: A comprehensive foundation. Neural Networks 2(2004), 41 (2004)
7. Jacobs, R.A.: Increased rates of convergence through learning rate adaptation. Neural networks 1(4), 295–307 (1988)
8. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
9. LeCun, Y.: The mnist database of handwritten digits. http://yann. lecun. com/exdb/mnist/ (1998)
10. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521(7553), 436–444 (2015)
11. Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., Khudanpur, S.: Recurrent neural network based language model. In: Interspeech. vol. 2, p. 3 (2010)
12. Mitchell, T.M.: Machine learning. McGraw Hill series in computer science, McGraw-Hill (1997)
13. Phansalkar, V., Sastry, P.: Analysis of the back-propagation algorithm with momentum. IEEE Transactions on Neural Networks 5(3), 505–506 (1994)
14. Plagianakos, V.P., Magoulas, G.D., Vrahatis, M.N.: Deterministic nonmonotone strategies for effective training of multilayer perceptrons. IEEE Transactions on Neural Networks 13(6), 1268–1284 (2002)
15. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological review 65(6), 386 (1958)
16. Umbaugh, S.E.: Computer vision and image processing: A practical approach using CViptools with Cdrom. Prentice Hall PTR (1997)
17. Yegnanarayana, B.: Artificial neural networks. PHI Learning Pvt. Ltd. (2009)