Highly Constrained Neural Networks for Industrial Quality Control
Nicola Guglielmi, Roberto Guerrieri, and Giorgio Baccarani

Abstract—In this work we investigate techniques for embedding domain-specific spatial invariances into highly-constrained neural networks. This information is used to drastically reduce the number of weights which have to be determined during the learning phase, thus allowing us to apply artificial neural networks to problems characterized by a relatively small number of available examples. As an application of the proposed methodology, we study the problem of optical inspection of machined parts. More specifically, we have characterized the performance of a network created according to this strategy, which accepts images of parts under inspection at its input and issues a flag at its output which states whether the part is defective. The results obtained so far show that the proposed methodology provides a potentially relevant approach for the quality control of industrial parts, as it offers both accuracy and short software development time, when compared with a classifier implemented using a standard approach.

I. INTRODUCTION

VISUAL inspection of industrial parts is a powerful technique for quality control. In fact, it allows to perform a cheap and nondestructive inspection of the materials surface revealing the presence of structural defects which must be trapped during the early stage of the manufacturing process.

A serious drawback of this technique is related to the significant cost of the software which must interpret the images. This high cost is related to a few factors. First of all, most inspection tasks need highly specialized solutions. This requirement accounts for the low volume of diffusion of each detection system, increasing the cost of the software shipped to the final customers.

A second consideration concerns the high reliability required to the detection system, joined with real-time operating conditions. This requirement is mirrored in the high quality of the software and in a very accurate phase of testing and optimization of every segment of the code. Since a large part of the required code has to be finely tuned to the specific task at hand, the development cost cannot be reduced by splitting it among different applications. For this reason, the learning capabilities of neural networks [1], [2] can make it feasible new applications, otherwise hampered by the high development cost.

Fluorescent magnetic particle inspection (FMPI) is a nondestructive method which permits a quality control in ferromagnetic materials. The resulting image, however, is complex and requires further processing for its interpretation. Fig. 1 shows an example of these images. Observe that a large amount of noise covers the presence of cracks. Simple thresholding of the image does not help removing the noise. In fact, many other artifacts have higher luminosity and similar shape and thus would not be discarded by such a simple procedure. For this reason, a more sophisticated technique is required to capture the essential details of the defects.

A popular approach is based on the Hough transform (HT) method [4], which models the patterns under investigation with a small number of suitable features. Line finding is a typical application of HT, which usually exhibits very good performance. The typical parameterization of a straight line is obtained using the variables \((\rho, \theta)\), where \(\rho\) is the module of the normal vector from the origin to the line and \(\theta\) is the angle between this vector and the \(x\)-axis. The presence of a line is then suggested by the detection of many input points associated with the same parameterization \((\rho^*, \theta^*)\). Apart from the complexity of this transform in terms of computational time, a direct inspection of a typical image (see Fig. 1) shows that many linear objects can be extracted from the scene, thus giving a modest selectivity of the HT method in this application. Experimental results obtained by using an HT-based strategy for the detection of cracks are presented by Raffo et al., in [5]. The obtained recognition ratio is lower, however, than the one measured in this work.

A common problem arising when neural networks are applied to image classification is due to the large dimensionality of the input space. If we consider a standard 740 \times 560 pixel image and connect each pixel to the input layer of a multilayer perceptron, the number of required weights would grow very high, increasing the required computational resources and the classification errors [6], [7]. The situation is worsened by the insufficient number of examples available for the training of the network. A few approaches have been proposed to reduce the above problem. The common background of
these techniques is the inclusion into the learning algorithm of constraints which reduce the degrees of freedom of the network. This amounts to introducing a bias into the solution space available to the classifier [6]. Since the bias strongly influences the behavior of the network, it is important to optimally design it. For instance, when no information about the problem is available, standard techniques introduce a "preference" for smooth solutions or try to minimize the number of weights required to perform a classification. A different situation arises when some domain knowledge is available about the specific task. In this case, a powerful approach can be devised by structuring the system so that constraints are directly embedded into it.

The adaptive methodology proposed in this paper attacks the design of the bias by exploiting the unstructured knowledge provided by human operators and designing the classifier so that the structure of the network already represents the available information, letting the learning algorithm perform the fine tuning. In this way, software development time can be drastically reduced when compared to a custom approach, and at the same time the resulting classifier is highly constrained, with few weights which must be determined by the learning procedure. Some preliminary results of this research were presented in [10], producing the first application (as far as we know) of a neural network performing a complete cycle of visual inspection of machined parts (image in, classification flag out).

Finally, we compare the adaptive technique with a customized texture analysis program based on statistical pattern recognition techniques [20] using several objective functions. Among these, the shorter development time implies that the proposed approach has a relevant impact on the cost structure of the application, while the accuracy is similar in the two cases.

The paper is organized as follows. In Section II, a concise description of the application is given. In Section III, the constraints used in this work are presented. In Section IV, a suitable software metric is introduced. In Section V, the dedicated processing is schematized and the neural algorithm is described. In Section VI, the results of the methodology are discussed in terms of error rate and robustness against reduced precision, and a comparison between the neural and the dedicated solution is given. Finally, in Section VII some conclusions are drawn.

II. THE APPLICATION

FMPI is a nondestructive method for quality control of ferromagnetic materials [3], [12]. The principle on which MPI methods are based consists in the leakage of magnetic flux in proximity of surface discontinuities. To apply this methodology, the metallic parts are covered by a solution of magnetic particles. Then, an external magnetic field, stronger than the intrinsic one, is applied. This field can be easily induced by forcing an electric current through a couple of electrodes connected to the metallic object. The induced field causes a redistribution of the magnetic powder with a maximum density near surface discontinuities. If the magnetic particles are fluorescent, a way to make these structural properties evident is the acquisition of the image using ultraviolet light. The resulting image shows the patterns induced by the magnetic field on the powder, but it is very complex, especially for some geometries of interest [13], [14] and requires further processing to lead to an interpretation.

An interesting "mechanical" preprocessing is suggested by Cheu [15] in his project of an automated connecting-rod crack detection system and consists of a controlled rinsing of the part to weaken the effects of edges which are clearly false defects. Unfortunately, in the presence of complex geometries, this mechanism is not reliable enough and the required rinsing is difficult to accommodate in several manufacturing steps. For this reason, a robust algorithm overcoming the previous limitations would provide a more effective inspection technique. The system we propose does not require any mechanical preprocessing of the image and thus is easier to use in a standard environment.

III. THE CONSTRAINTS

The development of the system began by interviewing experts. Interestingly, many rules were stated in terms of what is not relevant to the task at hand, thus suggesting to recast this information in terms of a suitable set of invariances. A first hint provided by the experts is that they are able to detect a defect looking at a portion of the image. A qualitative bound on the required image size is about 40 × 40 pixels. This implies that it is possible to examine relatively small portions (possibly overlapping) of the image in parallel. Another powerful constraint is that since the defect is a crack, its shape is roughly one dimensional. Even though its length can be substantial with strong variations, its thickness does not exceed a maximum of 5–9 pixels in our camera setup. Furthermore, direct inspection of the cracks shows that their local structure is approximately linear, even though the noise can introduce relevant distortions to their aspect. The above consideration suggests the use at the lowest level of the system of local feature detectors, associated with square blocks having a side somewhat larger than the typical thickness of a crack. These blocks, possibly overlapping, cover the portion under consideration of the image. The output of the feature detectors can then be fed to a network which combines all the available outputs to provide the final classification. In this framework, spatial invariances directly constrain the mutual relationships among the values of the weights of the feature detectors. More specifically, the classifier, and thus the feature detectors, should be invariant to translations, rotations, and chirality inversion of the image, while scale invariance is not applicable to this problem. The first two invariances, derived from a natural principle of isotropy, have been investigated by others (e.g., [16]), obtaining networks too complex for our application. More manageable networks implementing only rotation invariance [17] and approximate translation invariance [18] have been presented, while chirality invariance has not received any attention yet.

The main methods to obtain a network with a behavior invariant to a set of spatial transformations [16] are based either
on a suitable training procedure, or on a choice of an input data representation which embeds into itself the invariances or, finally, on the determination of a proper network topology. The first choice implies a learning of invariances by examples. It is possible only when a large number of training samples is available. Since the information about the invariances is not exploited explicitly, the size of the network remains very large, leading to high-variance solutions. The second choice involves the definition of representation systems of the input data having the invariance built in. For example, a polar parameterization of linear patterns automatically provides rotation invariance to the following stages of a network. This strategy is very powerful, but generally requires high computational effort to project the input data onto the invariant features space. The third technique is based on the possibility of designing a network topology embedding the required invariance. This technique has been used in this work since it substantially reduces the number of free parameters of a network and lets the learning procedure determine the best parameters required by the process. The solution space is thus heavily biased by the availability of a priori information, allowing the use of small databases to train the network and reducing the processing time for on-line applications.

IV. THE SOFTWARE COMPLEXITY MODEL

The information obtained by experts may be used either to draw up a dedicated and optimized program to reveal the presence of defective parts or for building the structure and the training set of a neural network. To compare these solutions it is necessary to analyze both the accuracy of the classifiers and the cost [8] of the software required to implement them.

Hence, in this section we introduce a set of parameters to describe and compare two software projects we have developed to have experimental data on the above subjects. This set comprises the number of personmonths required to develop the code $C$, the reliability $R$ of the code, and two additional parameters of interest, i.e., flexibility $F$, accounting for the incremental cost required when the existing software is used to solve a similar problem, and, finally, classification efficiency $P$.

We relate the previous parameters to the following quantities: the number of lines of code $n$, the number of lines of code $v$ developed during the preliminary investigation phases, the percentage of code $p$ to be modified when applying the detection system to similar problems, and the recognition ratio $r$, measured on the test set. In this context $r$ is the ratio between the number of misclassified images and the total number of images in the test set.

The following dependencies are assumed [9]

\begin{align}
C & \propto n + v \\
R & \propto \frac{1}{n} \\
F & \propto 1 - p \\
P & \propto r. \quad (1)
\end{align}

V. THE METHODOLOGIES

The software system operates on a data stream of 10 gray-scale images per second, with a resolution of 740 x 560 pixels (see Fig. 1). A detailed description of the problem can be found in [10].

The methodologies we consider correspond to two different strategies to solve the problem: the first one reflects an adaptive approach, while the second one identifies a substantially dedicated approach.

A. The Adaptive Solution

The classifier examines subimages having a size of $L \times L \sim 40x40$ pixels and consists of three layers. We now describe the architecture of each layer of the network, starting from the input.

1) The Feature Detectors: The units in the first layer act as feature detectors tuned to the shape of the defects. For the sake of simplicity, in this paper we describe an architecture with a single family of feature detectors. The general form of the feature detector $F$ we consider is

\[ F(j, i) = \sum_{(u, v) \in S} F_{ji}(u, v) \cdot I(j - u, i - v) \quad (2) \]

where $I$ is the input image, $(j, i)$ represents the address of a generic pixel, and $F_{ji}$ is a discrete convolution kernel related to pixel $(j, i)$ having compact support $S$. If several detectors are used at the same time, each one of them must satisfy the constraints discussed in the previous section, producing a linear increase of the number of independent weights. We shall now describe how the different invariances constrain the values of the weights of the feature detectors.

a) Translational invariance: Each unit covers a square patch of the image having size $N \times M$ (see Fig. 2) of the same order of the thickness of the crack (about 5–9 pixels in our implementation) and shares the value of its weights with all the other units, thus achieving positional invariance [18] and providing a strong reduction of free parameters. These patches, possibly overlapped, completely cover the image. That means that (2) reduces to

\[ F(j, i) = \sum_{n=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} \sum_{m=-\lfloor M/2 \rfloor}^{\lfloor M/2 \rfloor} F(n, m) \cdot I(j - n, i - m) \quad \forall (j, i) \quad (3) \]

where $F$ is independent of $(j, i)$. $N$ and $M$ are chosen as odd numbers for reasons of central symmetry and quotients $N/2$ and $M/2$ are integer divisions, i.e., $N/2 = (N - 1)/2$, and similarly for $M$.

b) Structural invariance: The number of weights can be further reduced by observing that a locally linear structure is sought. Hence, the weights must share the same values along the direction orthogonal to the crack. This reduces the number of independent weights by a factor $N$. The dimension of the processed patch, however, remains $N \times M$ to filter out the noise of the image. This is carried out by the low-pass behavior of the detector along the direction parallel to the crack. Thus,
rewriting $F(x, y) = C \times \mathcal{F}(y)$, we obtain from (3)

$$F(j, i) = C \sum_{n=-(N/2)}^{N/2} \sum_{m=-(M/2)}^{M/2} \mathcal{F}(m) \cdot I(j - n, i - m).$$

(4)

c) Chirality invariance: Finally, chirality invariance requires that the weights associated to the pixels on the left part of the patch have the same value of those on the right part of the patch. Thus, the total number of independent weights required by the feature detectors decreases now to $(M/2) + 1$ (five in our implementation). More formally, we obtain

$$F(j, i) = C \sum_{n=-(N/2)}^{N/2} \sum_{m=0}^{M/2} \mathcal{F}(m) \cdot [I(j - n, i - m) + I(j - n, i + m)].$$

(5)

d) Rotational invariance: Approximate rotational invariance is obtained by scanning multiple directions along the same image. This process can be formally described considering a number $D$ of images $I_{\alpha_d}$, $d = 1, \cdots, D$ derived from the same scene through rotations of angles $\alpha_d$. A possible choice for $D$ is

$$D = 2^k \quad k = 1, 2, \cdots, K.$$  

In our experience, four images rotated at 45 degrees, giving $K = 2$, reasonably approximate rotational invariance. We can associate to every image $I_{\alpha_d}$ a network, whose structure is independent from the processed direction and sharing its weights with the networks associated with the other directions. Note that a better angular resolution increases the computational load, while the number of independent weights remains the same. Hence, the number of operations increases linearly with the angular resolution, while the robustness of the classifier, determined by the number of free weights, is not affected.

2) Upper Layers of the Network: The upper layers of the network gather the evidence provided by the feature detectors and perform the final classification of the image. Since we are seeking an almost linear structure, the layer collecting the output of the feature detectors is organized as a set of $D$ groups of winner-take-all (WTA) networks, one for each angle $\alpha_d$. Considering a direction parallel to the columns of matrix $I_{\alpha_d}$, the associated group of WTA’s selects for each row of matrix $F_{\alpha_d} = F * I_{\alpha_d}$ ($*$ stands for convolution operator) the maximum of the values and stores them in a vector $V_{\alpha_d}^{max}$ (see Fig. 3).

$$V_{\alpha_d}^{max}(l) = \max_{i=1, \cdots, L} \{F_{\alpha_d}(l, i)\} \quad l = 1, \cdots, L.$$  

Note that given a certain direction $\alpha_d$ and the related image $I_{\alpha_d}$, the associated feature detector is characterized by a structural symmetry axis parallel to the direction under investigation. Thus, the following group of WTA’s processes the filtered image $F_{\alpha_d}$ along a direction orthogonal to the one scanned by the feature detector.

Each WTA is then organized as a binary tree of modules computing the “soft” maximum of two inputs values. To use the error backpropagation technique [1], the following differentiable soft maximum function has been used:

$$\text{softmax}(x, z) = \frac{1}{k} \log [\sigma(kx, kz)] + z$$

$$\sigma(x, z) = \frac{1}{1 + \exp(z - x)},$$

(6)

whose steepness can be tuned by changing the value of $k$. Note that this step does not introduce any free parameter. The final classification is then performed by a set of sigmoidal units in the third layer, associated to each vector $V_{\alpha_d}^{max}$ produced by the previous blocks. Each unit at this level estimates the norm $N(\alpha_d)$ of the related vector $V_{\alpha_d}^{max}$ and produces at its output.
Fig. 4. Classification layer (for a single direction $\alpha_d$).

![Diagram of classification layer]

Fig. 5. Final classification stage.

![Diagram showing logical OR and classification flags]

A flag $C(\alpha_d)$ (see Fig. 4)

$$C(\alpha_d) = \sigma[N(\alpha_d) - \Theta]$$  \hspace{1cm} (7)

where $\Theta$ is the neuron threshold learned during the training phase and $\sigma$ is a sigmoidal function. If at least a flag $C$ is issued, then a crack is detected in the image. The final classification stage is shown in Fig. 5, where $N$ represents the network previously described. Since all units have a differentiable transfer function, standard error back propagation can be used to determine the weights and thresholds of the network.

This classifier has been implemented using Aspirin [19], a NN development tool.

B. The Dedicated Solution

The second approach, which is here shortly described, splits the visual pattern analysis into five sequential phases shown in Fig. 6. Every processing phase reflects an accurate analysis of the specific application and involves a fine manual tuning of the parameters. For example, the choice of the filtering masks, performed in step 1, balances the impact of the convolution kernel on the classification accuracy and the execution time, forcing the kernel coefficients to be a power of two. This feature reduces the CPU time on cheap machines without hardware multipliers.

![Diagram showing processing flow]

The segmentation step is based on a double threshold hysteresis algorithm and the morphological analysis consists in a distance transform followed by a medial-axis transform [11]. The considered features of a pattern are both geometrical measures, like circularity and skeleton linearity, and texture-based, like moments and brightness measures. The first four phases extract the relevant parameters of the image, while the last step applies a classifier, based on a linear perceptron, to perform the final detection. The code has been developed in C language.

VI. RESULTS

A. Performances of the Neural Network

Since the patterns to be classified are subimages having size $40 \times 40$ pixels and the images have a size of $740 \times 560$ pixels, we obtain a partition of every image into 1036 patterns, if we consider an overlap coefficient of $1/2$ in both directions. We trained the classifier on a set of 200 independent subimages. These patches were extracted from 21 images, randomly picked out from a group of 42 images. Among them, three images presented cracks, while eighteen did not show defects. Ten subimages monitoring all of the observed defects were expressly extracted from the subgroup of defective images. Furthermore, 10 nondefective patches, considered difficult to recognize, were suggested to us by experts, and finally, from every one of the 18 images belonging to the normal subgroup, 10 patches have been randomly extracted. In conclusion, the training set has been formed by 10 patterns showing cracks and 190 patterns associated to a normal surface.
The large number of examples associated with good parts reflects the relative scarcity of production defects. Since in this application the cost of not detecting a crack is potentially high, however, we made the examples of the crack in the training sequence as frequent as the examples of the normal parts. This procedure amounts to the use of a suitably weighted Bayes decision rule [20].

Another group of 21 images has been used as test set: it is composed of images, four showing defective parts and 17 associated to normal cases. We tested the performance of the classifier on a set of 3000 patterns extracted from the remaining images. These patterns included 30 subimages monitoring all the defective parts and a number (100) of insidious patches suggested by the experts.

The values of $N(\alpha_d)$ provide an estimate of the likelihood of having a defect aligned along a direction orthogonal to $\alpha_d$. A typical output is shown in Fig. 7, where the probability of having a defect has been encoded using gray levels.

Note that only the real crack has been highlighted, even though other structures with higher brightness and elongated shape are visible in the scene. Fig. 8 shows that the weights learned by the feature detectors capture the shape of a crack. The obtained filter exhibits a differential behavior, showing that high frequency components of the analyzed signal are particularly meaningful.

The CPU time required by the learning phase is about five minutes on a SPARC2. Such a short CPU time has been obtained since the number of weights to be learned is extremely small. Defining now the relative error as the ratio between the number of misclassified patches and the total number of patches belonging to a given set, another important finding is that the error on the test set does not increase when the error on the training set is made very small, as shown in Fig. 9 where the error over the training set is zero at 20000 iterations. This implies that the network does not show the overfitting problem typical of redundant classifiers [6]. Another important point is that all the errors we have observed were "false positive" ones. We think that this is due to the consistency of the invariances that are directly linked to the physics of the problem. We have then investigated the effect of the reduction of the gray levels. Fig. 10 shows that the error does not appreciably increase by reducing the number of bits used to code the gray levels of the image down to three bits.

Finally, we have studied the effects of a lower resolution in the frequency domain. This experiment has been carried out by defocusing the lens of the camera. This is equivalent to a low-pass filtering of the image. The loss of the high frequency components introduces a serious degradation of the classifier performance, confirming that it behaves as a frequency-tuned system. Note, however, that a blurred image is not usual in...
problems, like those characterized by variations of the texture providing an outstanding recognition ratio. On the other hand B. The Software Comparison

industrial applications where the camera stands still and its setup can be calibrated with high precision.

The main advantage of the dedicated approach consists in the very high level of accuracy of the detection system, providing an outstanding recognition ratio. On the other hand the adaptive solution allows an immediate treatment of similar problems, like those characterized by variations of the texture of the patterns or by variations of the geometric scale, with the only effort of a rerun of the learning algorithm. The same task would instead require a revision of part of the code of the dedicated system.

The quantitative analysis of the two software tools has produced the results shown in Table I.

The measured recognition ratio \( r \)—defined as the ratio between the number of images containing at least one misclassified patch and the number of images in the set—is affected by errors of “false-positive” type. Assuming that an operator can double check the few positive cases, these errors could be easily filtered out. The percentage of lines of code modified to solve a similar problem, \( p \), has been quantified when the statistical properties of the background of the image are changed from the original case.

Considering the proportionality coefficients in (1) equal to one, we obtain the values of the objective functions associated to the two approaches as shown in Table II.

As a final data in Table III we report the development time for the two projects.

To conclude we can assert that—in general—the adaptive methodology provides an excellent trade-off between development cost and recognition accuracy, and only when the latter dominates all the other aspects is the dedicated solution worth pursuing.

### Table I

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Dedicated</th>
<th>Adaptive</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>850</td>
<td>90</td>
</tr>
<tr>
<td>( r )</td>
<td>420</td>
<td>420</td>
</tr>
<tr>
<td>( p )</td>
<td>30%</td>
<td>0%</td>
</tr>
<tr>
<td>( r )</td>
<td>95%</td>
<td>90%</td>
</tr>
</tbody>
</table>

### Table II

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Dedicated</th>
<th>Adaptive</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C )</td>
<td>1270</td>
<td>510</td>
</tr>
<tr>
<td>( R )</td>
<td>0.0012</td>
<td>0.011</td>
</tr>
<tr>
<td>( P )</td>
<td>0.7</td>
<td>1</td>
</tr>
<tr>
<td>( P )</td>
<td>0.96</td>
<td>0.9</td>
</tr>
</tbody>
</table>

### Table III

<table>
<thead>
<tr>
<th>Phase</th>
<th>Dedicated</th>
<th>Adaptive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Startup</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Coding</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Testing</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

### VII. CONCLUSION

In this paper we have illustrated a technique to include a priori information in a neural network. In our application, most of the information has been stated in terms of spatial and structural invariances and has been used to bias the network toward an acceptable solution. A remarkable reduction of the number of free weights has been achieved, thus reducing the CPU time required to learn the optimum values and increasing the overall robustness of the system. The method has then been applied to an industrial problem of image classification for quality control leading to a complete input–output processing system and showing at the same time robustness and computational parsimony.

Furthermore, the adaptive approach has been compared to a dedicated solution. Results show that an adaptive methodology gives important advantages in terms of software reusability and terseness. On the other side, reasons in favor of a dedicated approach emerge when an extremely high level of accuracy has to be granted.

### ACKNOWLEDGMENT

The authors acknowledge Dr. M. Mastretta and Dr. L. Devena of Automas S.C.R.I. for providing the database of images.

### REFERENCES