Multi-Scale Pyramidal Pooling Network for Generic Steel Defect Classification

Jonathan Masci and Ueli Meier and Gabriel Fricout and Jürgen Schmidhuber

Abstract—We introduce a Multi-Scale Pyramidal Pooling Network tailored to generic steel defect classification, featuring a novel pyramidal pooling layer at multiple scales and a novel encoding layer. Thanks to the former, the network does not require all images of a given classification task to be of equal size. The latter narrows the gap to bag-of-features approaches. On various benchmark datasets, we evaluate and compare our system to convolutional neural networks and state-of-the-art computer vision methods. We also present results on a real industrial steel defect classification problem, where existing architectures are not applicable as they require equally sized input images. Our method substantially outperforms previous methods based on engineered features. It can be seen as a fully supervised hierarchical bag-of-features extension that is trained online and can be fine-tuned for any given task.

I. INTRODUCTION

Automated Inspection Systems in steel industry aim at recognizing material defects to improve product quality and reduce production costs and human intervention. This is perhaps one of the most crucial phases of the whole quality control pipeline. Over the years, many efforts have been devoted to crafting features to solve this problem. Latest advances in technology suggests new acquisition systems (such as high-resolution and hyper-spectral cameras), making conventional features perform poorly. Handcrafting a new set of features plus corresponding parameters is a cumbersome process which may cost several man years and lots of money. It might be virtually impossible where the engineers have little prior knowledge, e.g., in the case of hyper-spectral acquisitions.

It is then paramount to have a self-adjusting system requiring only minimal efforts for manual selection of parameters. This issue is addressed widely in the literature. Below we briefly review the best frameworks in computer vision (CV) and machine learning (ML), respectively.

Many CV systems adopt the bag-of-features (BoF) approach [1]. For a given image a set of features (e.g. SIFT descriptors [2]) are extracted and then encoded in an overcomplete sparse representation using a dictionary-based technique. This produces a histogram of “active” words representative of the content of the image. The feature extraction is hard-coded whereas the dictionary is specifically built for a given task. That is, for classification tasks feature vectors from all or a subset of images from the training set are collected and clustered. The cluster centers form the basis of the dictionary that is used in the coding stage. Finally a supervised classifier is trained to classify the histogram representation of the image. This approach does not discover new and possibly very discriminative features and uses conventional algorithms. The effort is all in the encoding stage where a per-task dictionary of words is learned.

Supervised ML methods, on the other hand, try to map the pixel-based representation directly into a label vector, learning both feature extraction and encoding from a labelled dataset. Learning the features from the data helps to make the system easily applicable to domains where prior knowledge is not well consolidated, in particular in the case of textured steel, where the distinction between acceptable and defect patterns is hard even for the most advanced and consolidated systems. Perhaps the most successful methods for image classification are variants of Convolutional Neural Networks (CNN) reminiscent of simple and complex cells in the primary visual cortex [7]. Learning good features with such models, especially in cases where the number of training samples is scarce, opened up the investigation of unsupervised algorithms which can be used as a pre-training stage. This approach has become quite popular and is widely used [8], [9], [10], [11] to obtain better feature extractors in such setups. Whether pre-training leads to improved recognition accuracy for classification tasks is questionable though [12]. As long as a labelled dataset is available, fully supervised approaches seems to be preferable in many benchmarks [13]. For handwritten characters, elastic distortions and deformations are the best way to avoid overfitting and improve generalization [14], [15], [13]. CNN have been recently shown [16] to outperform any conventional feature for the task of steel defect classification, the only major limitation being the partitioning and resizing of the data to fit the constraint on equally sized input images. This requires several networks to be trained on a subset of the data, which reduces considerably the number of training patterns per class and makes training almost impossible in domains where labelled data are expensive, requiring then systems able to learn just from few samples per class.

Furthermore, the desired invariances for general steel defect recognition are not that easily synthesized—most of the defects cannot be detected by rotation invariant descriptors, and in most cases the scale is of crucial importance, too. A distortion/deformation-free approach is preferable.

There are obvious similarities between BoF and fully supervised CNN. Both extract features based on photometric discontinuities (e.g., edges), either engineered or learned from samples followed by an encoding stage. Standard CNN
lack multiple resolution pooling and the explicit encoding steps of winning BoF approaches. BoF, however, lack tunable feature extraction stages that may make complex encodings less important. A main drawback of CNN is their restriction to constant size input images, which the steel industry cannot simply overcome by resizing and padding.

A. Contributions

Here we present the MSPyrPool framework which aims at solving the general steel defect recognition problem. We further close the gap between BoF and CNN by introducing an extension of commonly used convolution-based nets for the steel industry, consisting of three new ingredients:

- a back-propagation-compatible Pyramidal Pooling layer which produces a fixed-dimensional feature vector independent of the input image size;
- a learnable encoding layer to incorporate commonly used encoding strategies of CV;
- a Multi-Scale feature extraction strategy.

Our approach is the first applicable to general steel defect classification problems with arbitrarily sized images. It scales straightforwardly to multi-variate (hyper-spectral) images, the coming standard in automatic steel inspection.

II. RELATED WORKS

Object recognition algorithms usually use the following scheme: feature extraction, encoding, classification. In CV is desirable to obtain a linearly separable code that can be effectively classified with a linear SVM for scalability reasons. This does not affect ML methods as the prediction comes at a linear cost of the forward pass of the network. In what follows we briefly recall first the CNN general architecture and then the two main concepts of a BoF system which will be reinterpreted in our MSPyrPool architecture.

A. Convolutional Neural Networks

CNN are hierarchical models alternating two basic operations, convolution and subsampling, reminiscent of simple and complex cells in the primary visual cortex [7]. They exploit the 2D structure of images via weight sharing, learning a set of convolutional filters. This powerful characteristic makes them excel in many object recognition [17], [6], [15], [16] and segmentation [18], [19] benchmarks. CNN are composed of the following layers:

- **Convolutional Layer**: convolves the set of input images \( \{f_i\}_{i \in I} \) with a bank of filters \( \{w_k\}_{k \in K} \), producing another set of images \( \{h_{ij}\}_{j \in J} \) denoted as maps. A connection table \( CT \) specifies input-output correspondences (inputImage \( i \), filterId \( k \), outputImage \( j \)). Filter responses from inputs connected to the same output image are linearly combined. This layer performs the following mapping:

\[
    h_{ij}(x) = \sum_{i,k \in CT,l,k,l} (f_i * w_k)(x),
\]

where * indicates the 2D valid convolution. Each filter \( w_k \) of a particular layer has the same size and defines, together with the size of the input, the size of the output maps \( h_j \).

The output maps are passed through a nonlinear activation function (e.g., \( \text{tanh}, \text{logistic} \), etc.).

- **Pooling Layer**: down-samples the input images by a constant factor keeping a value (e.g. maximum or average) for every non overlapping subregion of size \( p \) in the images. This layer does not only reduce the computational burden, but more importantly performs feature selection.

- **Fully Connected Layer**: this is the standard layer of a multi-layer network. It performs a linear multiplication of the input vector by a weight matrix.

Max-Pooling is our favored type of pooling, as it introduces invariances to small translations and distortions, and leads to faster convergence and better generalization [20]. The corresponding CNN are called Max-Pooling CNN, MPCNN for short. They are, so far, the best choice for a wide array of applications [17], [6], [15], [16].

B. Feature Encoding Algorithms

In the CV framework, features are extracted using engineered approaches (e.g. SIFT descriptor). What usually varies is not the feature extraction procedure but where in the image to extract the descriptors. The most successful methods extract features over a dense, equally spaced grid [21]. Recent improvements in classification performance on commonly used benchmarks [22] are rather due to improved encoding strategies than new feature descriptors. In order to produce a histogram, the descriptors need to be quantized such that they can be matched against a given codebook. This step is crucial to produce encodings with the right level of detail that avoids overfitting and leads to improved generalization to unseen data. The de-facto standard for this procedure seems to be given by overcomplete and sparse encodings of the feature descriptors. Most commonly used algorithms for this step are: Vector Quantization (VQ) where only one basis is selected; Sparse Coding (SC) where a small subset is kept; Locality-constrained Linear Coding (LLC) where the subset of basis, of fixed size, is selected using K-NN leading in fast coefficients estimation solving a very small least-squares problem.

C. Feature Pooling

Once the features are encoded a histogram is formed. The naïve approach, used in early BoF systems, is to sum all the \( N \)-dimensional codes, where \( N \) represents the number of bases in the dictionary, thus producing a global representation. A more powerful histogram generation technique is presented in [23], where features are considered in their spatial locality. At every level \( l \) of a quad-tree \( 2^l \) tiles are produced and for each tile a feature vector is extracted. This approach is used in the PHOG descriptor [21], an improvement over HOG [24]. In conjunction several methods to pool have been presented, such as sum-, average-, and \( \ell^2 \)-pooling.
CNN and BoF approaches share many building blocks. Both extracts features using convolutional filters, with the only difference being that the filters of a CNN are learned from the data whereas fixed feature extractors are used in BoF. Furthermore, BoF is inherently single layer, whereas deep multilayer architectures are capable of extracting more powerful features [25]. For HOG the image gradient is obtained using convolutional filters and then pooled to obtain the final descriptor. In SIFT an input patch of 64 × 64 pixels is tiled in 16 quadrants and for each of them a 8-dimensional vector of gradient orientation is extracted. The resulting descriptor is a concatenation of such vectors resulting in a 128-dimensional feature vector positioned at the centre of the input patch. If we interpret a CNN as a composition of many functions (e.g. layers), we obtain

\[ \text{CNN}(x) = f_c \ldots f_k \ldots f_1(x), \]  

(2)

where \( f_1 \) to \( f_c \) represent the feature extraction layers, \( f_{k} \) to \( f_1 \) the encoding layers and \( f_c \) the remaining classification layers. Providing a differentiable definition for each layer results in a framework whose parameters can be jointly learned from a labeled training dataset. In what follows we introduce two new layers that are used in our MSPyrPool framework.

A. Pyramidal Pooling Layer

In previous work [26] a dynamic pooling layer is used to obtain features independent of input size, to detect paraphrases of 1D signals. Here we present a variation which takes into account the 2D nature of images and produces spatially located representations at several resolutions following [23]. Let us consider a layer with \( k \) maps, each map corresponding to a filtered/pooled image. If all the values in each of those maps are summed, a \( k \)-dimensional feature vector which does not depend on the actual input image size, is obtained. This produces a representation that only depends on the number of maps producing a system which no longer requires fixed size images. This already represents a major improvement to obtain a generic steel defects classifier. However, summing all the activations of a map results in higher values for bigger images. To obtain a more stable measure, average pooling is usually preferred. An even more effective way of performing feature pooling uses the max operator instead of the average. This avoids normalization all together and in our experiments always speeded up learning. Hereafter we consider only pyramidal pooling layers with max-pooling.

To produce a spatially localized feature vector we divide the image in tiles and pool over the quadrants varying the pooling window accordingly. The final representation is then obtained concatenating the results of each of the levels, just as in BoF and is schematized in Figure 2. Using a sufficient number of tiles Pyramidal Pooling is equivalent to the conventional pooling operation of CNN and can therefore be considered as its generalization. Let us for example consider the case of \( 10 \times 10 \) images and max-pooling of \( 2 \times 2 \); this clearly equals to a Pyramidal Pooling with 5 tiles in each dimension.

The Pyramidal Pooling layer does not have any tunable parameters but in order to train the feature extraction layers before it the partial derivative of the layer’s output w.r.t. its input is required. Let us denote by \( \mathbf{X} \) the 3-dimensional input vector of images (e.g. [#rows, #cols, #maps]). During the forward pass max-pooling keeps only the maxima values in non overlapping sub-regions of \( \mathbf{X} \), down-sampling the images by a constant factor. The backward pass places the delta values (results of partial differentiation by applying the chain-rule) at the location at which the maxima was found, up-sampling to the original input size. In our pyramidal pooling layer the forward pass is equivalent to applying a subsampling operation at each level of the pyramid and then concatenating the result into a single output vector. Consequently the backward pass sums over the back propagation of each of the pyramid levels.

![Fig. 2. Pyramidal Pooling Layer. Features are pooled along \( l^2 \) equally sized quadrants and the histogram-like representations are concatenated to form a feature vector.](image-url)

B. Multi-scale extraction

A pyramidal feature extraction, while being itself already an improvement over conventional CNN, as it allows to relax the constraint on a fixed input size, extracts features corresponding only to a single scale (e.g. the nominal size of the “simplified” image where the pooling is performed). In many applications, where input images come at very different scales, applying a pyramidal pooling layer will not completely solve the problem. Multi-Scale pyramid feature extraction can be done using a pyramidal pooling layer for each representation (i.e. layer in the network), and then concatenating the various feature vectors for the classification stage. After a subsampling layer the image gets down-sampled by a constant factor and attaching a pyramidal pooling before and after a max-pooling operation therefore delivers a Multi-Scale feature extraction (Fig. 1).
C. Feature encoding layer

The next step in narrowing the gap between the conventional BoF approach and CNN is represented by introducing a feature encoding layer, the major contribution of this work. If we consider the simplest of the feature quantization algorithms of BoF, \( k \)-means with hard assignment, we note that such an algorithm can be approximated by a correlation based measure \(^1\). This allows us to derive a differentiable approximation of such coding scheme which we name as MLPDict.

1) MLPDict Layer: We use a fully connected layer with max-pooling to mimic the behavior of \( k \)-means coding. The projection of \( x \) with \( W \), the weights of the encoding layer, is the correlation between \( x \) and each column of \( W \). Taking the maximum approximates the VQ coding scheme. Compared to the BoF approach, \( W \) serves as the dictionary, however an adaptive one which is tuned sample after sample. Performing a pyramidal pooling operation on the result of such an encoding will produce a histogram-like representation.

Figure 3 shows a schematic representation of the encoding layer. The hidden representation of a convolutional-based encoding will produce a histogram-like representation. A pyramidal pooling operation on the result of such an encoding reduces to the correlation between \( x \) and \( y \), normalized to have zero mean and unit variance (\( \sum_i (x_i - y_i)^2 = \sqrt{\sum_i x_i^2 + \sum_i y_i^2 - 2 \sum_i x_i y_i} \)).

IV. Experiments

In all experiments, unless stated otherwise, we evaluate the average per-class accuracy to establish the classification performance, a more meaningful measure in case of unevenly distributed datasets. No additional preprocessing, such as translation or deformation is used because of the aforementioned requirements of steel industry. Of course any ad-hoc transformation of the data can be easily plugged in to improve performance. All nets are trained using stochastic gradient descent (mini-batch of 1) with initial learning rate of 0.001, annealed by a factor of 0.97 at every epoch and momentum term of 0.9. Best results are usually reached in few training epochs (\( \approx 30 \)), whereas for CNN many more are generally required, especially when transformations are added to the input. Softmax activation is used at the output layer and the Multi-Class Cross-Entropy loss is minimized. We validate our model first on publicly available benchmarks to compare with other published approaches. We then show results on a challenging dataset from the steel industry where our MSPyrPool framework can be applied directly and where CNN fail.

Comparing our architecture with CNN is not an easy task as the two approaches differ considerably. Nevertheless we try to make the comparison as fair as possible by taking equally sized convolutional and subsampling layers in both architectures and letting them differ for the choice of the encoding and classification stages. The two systems equals for a particular choice of the MSPyrPool parametrization.

GPU Implementation The proposed framework requires lot of computational resources to train. We provide a GPU implementation using Arrayfire [27] for which we experienced speed-ups in the range \( 15 \rightarrow 40 \times \), without loosing flexibility and ease of coding.

A. Conventional Benchmarks

We select three common evaluation datasets: digit, texture and object recognition. All of them belong to quite orthogo-
fig. 3. The MLPdict layer used for feature encoding. Image responses of a network layer are reshaped to produce \( D \) dimensional feature vectors, where \( D \) represents the number of images in the layer. Each pixel descriptor is mapped into another representation of size \( D' \) for which only the maxima value per row is preserved.

The MLPdict layer used for feature encoding. Image responses of a network layer are reshaped to produce \( D \) dimensional feature vectors, where \( D \) represents the number of images in the layer. Each pixel descriptor is mapped into another representation of size \( D' \) for which only the maxima value per row is preserved.

Nal domains for which ad-hoc techniques are usually applied. In particular the latter one shows clearly the advantage of a MSPyrPool network w.r.t. CNN in the context of fully supervised classification. No particular tuning of the architecture has been made as the aim of this section is to show the relative improvement of MSPyrPool nets over CNN.

1) **MNIST:** As a reference to compare our approach with a conventional CNN we take the well studied MNIST [4] benchmark of handwritten characters. CNN excel on this dataset where all digits are of equal size and centered in the middle of a \( 28 \times 28 \) grey-scale image. For this experiment the overall classification accuracy is used to easily compare with other methods. We use a MSPyrPool net with a convolutional layer with \( 5 \times 5 \) filters and 100 output maps (\( C \times 5 \times 5 \times 100 \)), a \( 2 \times 2 \) max-subsampling layer (\( MP \times 2 \times 2 \)), a pyramidal pooling with linear MLPDict at the output of the two subsampling layers with \( l = \{ 1, 2, 4, 8 \} \) and \( l = \{ 1, 2, 4 \} \) quadrants respectively. Results are shown in Table I. It is interesting to note that our approach is the best among the fully supervised CNN approaches and on-par with the ones which use unsupervised pre-training while just using a very small single layer network and 100 filters.

<table>
<thead>
<tr>
<th>Test %</th>
<th>CNN LeNet-5 [4]</th>
<th>99.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN + pre-training [28]</td>
<td>99.40</td>
<td></td>
</tr>
<tr>
<td>MSPyrPool</td>
<td>99.13</td>
<td></td>
</tr>
</tbody>
</table>

2) **CURET:** The Columbia-Utrecht (CURET) database [29] contains 61 textures; each with 205 images obtained under different viewing and illumination conditions. For training our architecture only a single image is required as input, just as in previous work [30], with no information (implicit or explicit) about the illumination and viewing conditions. We use the conventional evaluation protocol [30] but with a different random split of the data. Images are normalized to have zero mean and unit variance to compensate for very different light conditions. We train a CNN with 5 hidden layers: \( C \times 11 \times 11 \times 20, MP \times 5 \times 5, C \times 9 \times 9 \times 20, MP \times 5 \times 5 \), classification layer. We use \( \tanh \) as activation for every convolutional layer. We compare the result with a MSPyrPool with \( \tanh \) pyramidal pooling MLPDict and codebook size of 100 at the output of the first and second convolutional layers, keeping the rest of the network topology. Features are pooled using \( l = \{ 1, 2, 4 \} \) and \( l = \{ 1, 2, 3 \} \) levels producing a 3500 dimensional vector.

Table II compares our new MSPyrPool network with conventional CNN. The MSPyrPool net generalizes much better to the unseen test data and shows the superiority over conventional CNN for the task of texture classification, a domain closely related to steel classification. We also report results for the case where no MLPDict is used, to further show that the encoding stage is important and delivers indeed non-marginal improvements in final classification performance. The approach reaches 99.0% recognition rate on all 61 classes and thus greatly outperforms bank-of-filter and Texton (96.4% [30]), whose similar architecture has pre-wired feature extractors. So the novel layers help indeed to learn better features, and are a valuable contribution to the CNN framework.

<table>
<thead>
<tr>
<th>Test %</th>
<th>Textons [30]</th>
<th>96.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>96.5</td>
<td></td>
</tr>
<tr>
<td>MSPyrPool (no encoding)</td>
<td>93.8</td>
<td></td>
</tr>
<tr>
<td>MSPyrPool</td>
<td>99.0</td>
<td></td>
</tr>
</tbody>
</table>

3) **Caltech101:** A further validation of the proposed system is performed on a classical pattern recognition benchmark, Caltech101, where fully supervised CNN have seldom successfully been applied. Usually ad-hoc prepro-
cessing stages and tailored non-linearities are required to obtain a satisfactory performance. We compare our results with those obtained with the similar system which uses conventional BoF with VQ [23]. We use 30 images per class for training and we test on at most 50 of the remaining images converted to grey-scale. We consider a net with and without an encoding layer. Both nets are composed by: C\textsubscript{16} × 16 × 100, MP 5 × 5. For the net without an encoding layer a pyramidal pooling layer with \( l = \{1, 2, 4\} \) is used to create a 2100-dim feature vector. For the net with an encoding layer, we used a dictionary size of 1024 just before the pyramidal pooling layer. Using a pyramidal pooling layer with \( l = \{1, 2, 4\} \) results in a 21504-dim feature vector. We train both a net with a linear and non-linear activation function in the encoding layer. For the sake of completeness we also train a CNN consisting of: C\textsubscript{16} × 16 × 100; MP 5 × 5; C\textsubscript{13} × 13 × 100; MP 5 × 5; fully connected classification layer. This CNN architecture is by no means the best architecture for this task, and is only listed to quantify the improvement using MSPyrPool nets. Results of all experiments together with results from the literature are listed in Table III.

<table>
<thead>
<tr>
<th></th>
<th>Test %</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>25.2</td>
</tr>
<tr>
<td>net1</td>
<td>52.8</td>
</tr>
<tr>
<td>net2</td>
<td>58.0</td>
</tr>
<tr>
<td>net3</td>
<td>55.2</td>
</tr>
<tr>
<td>Spatial Pyramid [23]</td>
<td>64.6</td>
</tr>
<tr>
<td>LLC [22]</td>
<td>73.4</td>
</tr>
</tbody>
</table>

MSPyrPool nets clearly improve recognition rate compared to a similar sized CNN.\(^2\) Using an encoding layer improves generalization performance even though the resulting nets have many more free parameters. Results show that we are able to jointly learn the feature extraction, the quantization and the classification stages fully online in a particularly difficult domain where unsupervised pre-training is required in most ML systems. We also see that the encoding stage does not match the performance of the CV system, perhaps due to the single layer architecture adopted to mimic the one of SIFT descriptors.

B. Steel-Defects Industrial Benchmark

Steel is a textured material and defects come at varying scales, which makes the task of classifying a wide range of defects extremely difficult. It is not easy to find a good re-sizing technique without destroying the original information content of the images. As a matter of fact if images/objects in a given classification task are varying over a few order of magnitudes it is even impossible to resize or pad the images.

\(^2\)Better results are obtained with deeper and bigger CNN, we got 40\% with a huge CNN still worse than MSPyrPool.

In this context the established industrial pipeline involves a series of hand-crafted features which are hard to select and tune. This is particularly true for multi-variate images where prior knowledge is still not consolidated to produce an effective way of extracting steel features.

In [16] the authors show that features can be effectively and efficiently learned from raw-pixel intensities of steel defects using CNN, delivering online processing times and superior performance. Their approach inherits all the benefits of CNN and therefore extends well to any kind of input images and does not require any prior knowledge on the task. Unfortunately the intra-class defect variability makes the task of dividing the images into homogeneous size/ratio clusters cumbersome (defects within the same class can appear at very different sizes). The number of available samples for training would reduce dramatically. Using a MSPyrPool net avoids such problems because of the input-size independent feature extraction, one of the main contributions of this work, and makes convolution based networks applicable to the general steel defect recognition task for the first time. For this experiment we use a proprietary dataset of ArcelorMittal from a hot-strip mill production line containing 30 different defect classes. A region-of-interest (ROI) is provided for each of the instances which vary greatly in size from a minimum edge length of \( \approx 20 \) to a maximum of \( \approx 2000 \) pixels (Fig. 4). Furthermore the dataset is unevenly distributed w.r.t. the number of samples per class; a task where the full potential of our model can be appreciated. To obtain a good support to perform the pyramidal pooling we add background information to get a minimum patch size of 100 pixels along each dimension whenever possible and zero-pad otherwise. We also limit the maximum size per dimension to 500 pixels to accelerate training. We compare our system to a set of classifiers trained on commonly used features using the same evaluation protocol described in [16] but with a larger MLP for classification (500 units) as the task.

![Fig. 4. Subset of images from the Steel-defects benchmark showing the great difference in size among various samples. In this setting it is not possible to resize the images to the same size, hence a CNN is not applicable to solve this task.](image-url)
We use a MSPyrPool with: C corresponding classes (off diagonal elements in the confusion matrix). This

Fig. 5. Two misclassified images for which the network inverted the tanh for every convolutional layer. Pyramidal pooling layers with classifier by large margin.

...size information. We also see that the performance of our model, which learns everything from pixel representation with no prior knowledge, outperforms any engineered feature classifier by large margin.

V. CONCLUSIONS

We proposed an extension of CNN tailored to generic steel defect classification. It has three new ingredients: 1) a pyramidal pooling layer that makes the net independent of input image size; 2) multi-scale feature extraction; 3) an encoding layer emulating standard dictionary-based encoding strategies. We validated the novel architecture on various benchmarks, including a very challenging real industrial application, and obtained results comparable to or better than the current state-of-the-art. The full potential becomes evident when images for a given classification task vary in size—such problems so far have eluded standard CNN.

Our work closes the gap between neural network-based models and weak classifiers such as BoF. To the best of our knowledge, ours is the first architecture that learns both feature extraction and encoding online and in fully supervised fashion.

We believe that more complex encoding layers may considerably boost recognition rates of our Multi-Scale Pyramidal Pooling network, which we view as a new and flexible framework for supervised object classification.

REFERENCES


TABLE IV

CLASSIFICATION RESULTS FOR THE STEEL-DEFECT BENCHMARK WHERE CNN FAIL. VARIOUS CLASSIFIERS TRAINED ON CONVENTIONAL FEATURES ARE COMPARED WITH OUR MSPyrPool NETWORK. OUR METHOD OUTPERFORMS BY LARGE MARGIN ANY CLASSIFIER BASED ON ENGINEERED FEATURES.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Test %</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBP</td>
<td>29.1</td>
</tr>
<tr>
<td>LBP-HF</td>
<td>48.9</td>
</tr>
<tr>
<td>MONO-LBP</td>
<td>64.0</td>
</tr>
<tr>
<td>VAR</td>
<td>34.3</td>
</tr>
<tr>
<td>HOG</td>
<td>60.5</td>
</tr>
<tr>
<td>PHOG</td>
<td>58.9</td>
</tr>
<tr>
<td>CNN</td>
<td>N/A</td>
</tr>
<tr>
<td>MSPyrPool</td>
<td>75.3</td>
</tr>
</tbody>
</table>

is harder and the number of defect classes is much larger. We use a MSPyrPool with: C 11 × 11 × 20, MP 2 × 2, C 9 × 9 × 20, classification layer. tanh activation is used for every convolutional layer. Pyramidal pooling layers with tanh MLPDict and codebook size of 100 are attached at the output of the first and second convolutional layers. Features are pooled respectively at l = {1, 2, 4} and l = {1, 2, 3} levels producing a vector of 3500 dimensions.

Results are summarized in Table IV. We see that a MSPyrPool net can be applied to solve a problem where conventional CNN can not be applied. In this setting resizing will basically destroy the images and will remove the actual defect size information. We also see that the performance of our model, which learns everything from pixel representation with no prior knowledge, outperforms any engineered feature classifier by large margin.