Object Recognition with Multi-Scale Pyramidal Pooling Networks

Jonathan Masci[1], Ueli Meier[1], Gabriel Fricout[2], and Jürgen Schmidhuber[1]

[1] IDSIA – USI – SUPSI, Manno – Lugano, Switzerland,
jonathan@idsia.ch, http://idsia.ch/~masci/
[2] ArcelorMittal, Maizières Research, Measurement and Control Dept., France

Abstract. We present a Multi-Scale Pyramidal Pooling Network, 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 encoding layer improves generalisation performance in comparison to similar neural network architectures, especially when training data is scarce. We evaluate and compare our system to convolutional neural networks and state-of-the-art computer vision methods on various benchmark datasets. We also present results on industrial steel defect classification, where existing architectures are not applicable because of the constraint on equally sized input images. The proposed architecture 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.

Keywords: neural networks, pattern recognition, industrial application, bag-of-features, computer vision

1 Introduction

Localising and recognising generic objects within varying scenes is of crucial importance for computer vision and machine learning applications. The difficulty of this task is due to high view point-dependent variability and high in-class variability of certain objects. Nevertheless, recognition algorithms are already quite successful in recognising and localising objects. This success can be attributed to orientation and scale invariant feature extractors [1,2,3,4] or to dedicated neural network architectures, able to capture the two-dimensional structure of images and learn invariant representations [5,6,7,8].

In computer vision one winning strategy is based on the so-called bag-of-features approach [9,10]. For a given image a set of features, such as SIFT descriptors, are extracted and then encoded in an overcomplete sparse representation using a dictionary-based coding 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 centres 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.

Machine learning methods, on the other hand, try to map the pixel-based representation directly into a label vector, learning the feature extraction and coding from a labelled dataset. Among the most successful methods for image classification are variants of Convolutional Neural Networks [11,12,13,14] reminiscent of simple and complex cells in the primary visual cortex [15]. Learning good features for such models, especially in cases where the number of training samples is scarce, opened up the investigation of many unsupervised algorithms which can be used as a pre-training stage. This approach has been quite popular and is widely used [16,17,18,19,20] to obtain better feature extractors in such setups. It is very effective for clustering or dimensionality reduction [21] but whether such pre-training leads to improved recognition accuracy for classification tasks is questionable though [22]. As long as a labelled dataset is available, a fully supervised approach seems to be favourable and is the winning strategy in many benchmarks [23]. A main issue remains how to effectively avoid over-fitting and improve generalisation capabilities to previously unseen images, which in many cases might be very different from the training images. For handwritten characters elastic distortions and deformations are the best way to avoid over-fitting and improve generalisation properties [24,25]. The same approach has successfully been applied to Chinese characters [26]. For general object recognition tasks, the desired invariances are not that easily synthesised. Using affine transformations such as scaling, translation and rotations is the key to avoid over-fitting [27,28,14].

When it comes to more difficult tasks with the same object at different scales and varying location within the image, as in almost all problems of interest for computer vision, the bag-of-features approach often excels when histograms are produced at different levels [29]. Pyramidal pooling is nowadays the gold standard; recent improvements are rather due to new coding strategies. Most notably linear local coding (LLC) [30] led to improved results on various object recognition benchmarks.

There are obvious similarities between the bag-of-features approach and the fully supervised CNN. Both extract features based on photometric discontinuities (e.g. edges) either engineered or learnt from samples followed by an encoding stage. CNN lack of the multiple resolution pooling and moreover of an explicit encoding step which mimics the winning approaches of the bag-of-features methods. The latter lacks in a tuneable feature extraction stage which can leverage the need of a complex encoding step. Additionally CNN have the main drawback in being constrained to have constant input sized images, which can be overcome by resizing and padding, but still quite limiting in many applications.

In this paper we further close the gap between these two approaches and introduce an extension of commonly used CNN, consisting of three new ingredients: 1) the Pyramidal Pooling layer which produces a fixed-dimensional feature vector independent of the input image size; 2) a coding layer to incorporate commonly used coding strategies and 3) a multi-scale feature extraction to capture regularities visible only at certain scales.

2 Related Works

Almost all object recognition algorithms use the following scheme: feature extraction, feature encoding, classification. It is desirable to obtain a linearly separable code that can be effectively classified with a linear SVM for example. For real-world applications with a very large amount of data and many different classes [31,32] this is as a matter of fact paramount, since non-linear SVM classifiers are computationally just too expensive. In what follows we briefly recall the two main concepts of a bag-of-features system which will be reinterpreted in our Multi-Scale Pyramidal Pooling (MSPyrPool) architecture.

2.1 Feature Encoding Algorithms

In the computer vision framework, features are extracted using engineered approaches such as the widely used SIFT descriptors. What usually varies is not the feature extraction procedure per-se but where in the image to extract the descriptors. The most successful methods extract features over a dense, equally spaced grid [3]. Recent improvements in classification performance on commonly used benchmarks [30] are rather due to improved encoding strategies than new feature descriptors. In order to produce a histogram, the descriptors need to be quantised such that they can be matched against a given codebook (set of basis). This step is crucial to produce an encoding with the right level of detail that avoids overfitting and hence leads to improved generalisation to unseen data. The de-facto standard for this procedure seems to be given by over-complete and sparse encodings of the feature descriptors.

Here we briefly review three of the most commonly used algorithms for feature encoding, ranging from the simplest up to the current state-of-the-art method and later we show how these approaches can be used in the fully supervised framework of our MSPyrPool network.

1. **Vector Quantisation (VQ)**. In its original formulation, the spatial pyramid matching uses k-means to obtain a dictionary of centroids (clusters), $\mathbf{B} = [b_1, b_2, \ldots, b_N]$, which is then used to solve the following least squares problem

$$\begin{aligned} \underset{\mathbf{C}}{\operatorname{argmin}} &= \sum_i^N ||x_i - \mathbf{B}c_i||^2 \\ & s.t.\ \forall i.\ ||c_i||_0 = 1, ||c_i||_1 = 1, c_i \geq 0 \end{aligned} \tag{1}$$

   where the ℓ^0 constraint ensures that only one basis will be on and the ℓ^1 constraint ensures that its coefficient value will be 1. In practice a 1 is placed at the position of the nearest neighbour centroid, producing an approximation of x_i using a single code vector.

2. **Sparse Coding (SC)**. Using just a single basis vector results in high quantisation errors. Relaxing the ℓ^0 constraint in eq. 1 allowing more than one active basis reduces the quantisation error dramatically. Since the number of bases is bigger than the number of input dimensions the resulting system is underdetermined. An effective way to find a solution is to impose a sparsity regulariser, which reformulates the problem in a conventional and well studied sparse coding problem:

$$\underset{\mathbf{C}}{\operatorname{argmin}} = \sum_{i}^{N} ||x_i - \mathbf{B}c_i||^2 + \lambda ||c_i||_{\ell^1} \tag{2}$$

   This produces better reconstruction and together with a linear SVM outperforms non-linear approaches on the Caltech-101 benchmark [33].
3. **Locality-constrained Linear Coding (LLC)**. A more powerful quantisation algorithm [30] exploits the locality principle to obtain sparse representations as "locality leads to sparsity but not vice-versa". LLC solves the following equation:

$$\underset{\mathbf{C}}{\operatorname{argmin}} = \sum_{i}^{N} ||x_i - \mathbf{B}c_i||^2 + \lambda ||d_i \odot c_i||^2 \tag{3}$$

   where $\odot$ denotes the element-wise multiplication and d_i represents a regulariser to favour quantisation using bases similar to x_i. LLC is very effective in commonly used benchmarks such as Caltech-101, Caltech-256 and PASCAL VOC and can be computed very efficiently using only the k-nearest neighbours of the dictionary to reconstruct x_i instead of explicitly solving eq. 3.

2.2 Feature Pooling

Once the features are encoded a histogram is formed. The naïve approach, used in early bag-of-features 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 [29], where features are considered in their spatial locality and representations at different levels are extracted using a quad-tree. At every level l of a quad-tree 2^l tiles are produced and for each tile a feature vector is extracted. The same approach is also used to produce PHOG [3] descriptors, an improvement over HOG [2]. In conjunction with a pyramid of features several methods to pool over the quadrant have been presented, such as sum-, average-, and ℓ^2-pooling.

3 Multi-Scale Pyramidal Pooling Network

CNN and bag-of-features approaches share many basic building blocks. In both cases the feature extraction is performed using convolutional filters, with the only difference being that the filters of a CNN are learnt from the data whereas fixed filters (feature extractors) are used in bag-of-feature approaches. Furthermore, bag-of-features approaches are inherently single layer, whereas deep multilayer architectures consisting of many layers of feature extraction are capable of extracting more powerful features [34].

If we interpret a CNN as a composition of many functions (layers), we obtain

$$CNN(x) = f_c \dots f_k \dots f_e \dots f_1(x) \tag{4}$$

where f_1 to f_e represent the feature extraction layers, f_e to f_k 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 learnt from a training dataset.

In what follows we introduce two new layers that are used in our Multi-Scale Pyramidal Pooling Network.

3.1 Pyramidal Pooling Layer

Previous work [35] uses a dynamic pooling layer 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 [29]. Max-pooling further improves performance of the pooling operation. The last layer of the feature extraction part can be constrained to extract a histogram-like representation at multiple levels in the spatial pyramids. 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 simple idea produces a representation that only depends on the number of maps and which already represents a major improvement over conventional CNN.

However, summing all the activations of a map results in higher values for bigger images. To obtain a more stable measure, average pooling is preferred. An even more effective way of performing feature pooling uses the max operator instead of the average. This avoids normalisation all together and in our experiments always speeded up learning. Hereafter we consider only pyramidal pooling layers with max-pooling.

Still the resulting representation remains global. That is, a feature vector which is not able to capture the spatial locality of the features and their correlations within the various parts of an image. To alleviate these shortcomings the image is divided in a quad-tree fashion. A feature vector is extracted for each of the quadrants dynamically changing the pooling size according to the number of tiles which are required at each pyramid level. The final representation is then obtained concatenating the results of each of the levels, just as in the bag-of-feature approach. A graphical representation of the pyramidal pooling concept is shown in Figure 1 and closely follows [29].

The Pyramidal Pooling layer does not have any tuneable parameters but in order to train the feature extraction layers before the pyramidal pooling layer the errors need to be back-propagated (that is the partial derivative of the layer's output w.r.t. its input is required).

Let us recall how the conventional max-subsampling operation works and let us denote by $\mathbf{X}$ the 3-dimensional input vector (e.g. $[\#rows, \#cols, \#maps]$). During the forward pass only a value in every non overlapping sub-region of $\mathbf{X}$ is preserved. Hence the image gets down-sampled by a constant factor given by the pooling size (e.g. usually 2×2). The backward pass places the delta values (results of partial differentiation by applying the chain-rule) at the location at which the maxima value was found, producing an output sized as $\mathbf{X}$. In our pyramidal pooling layer, the forward pass is equivalent to applying a subsampling operation at each level in the pyramid, built over $\mathbf{X}$, and then concatenating the result into a single output vector. Consequently the backward pass corresponds to the sum of the back-propagation of each of the subsampling operations.

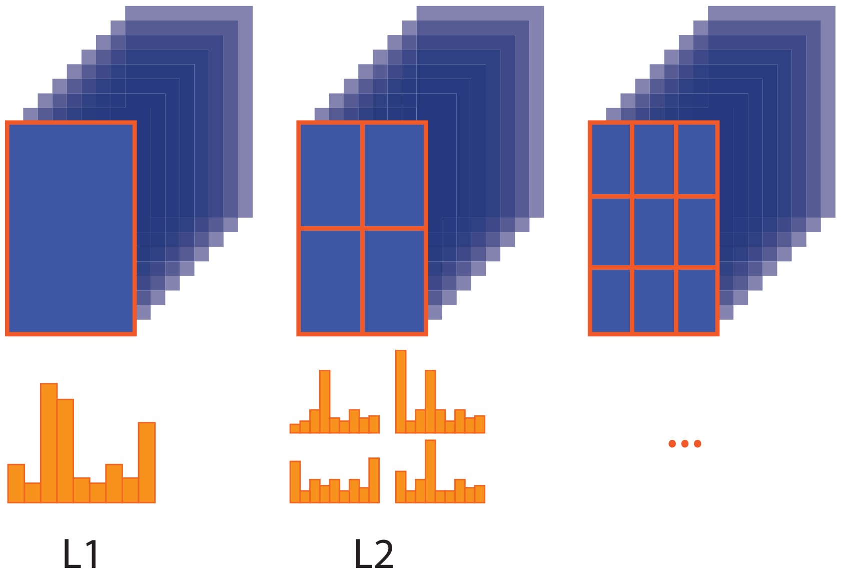

Fig. 1. Schematic representation of a Pyramidal Pooling Layer. The features are pooled along l^2 equally sized quadrants and the histogram-like representations are concatenated to form a feature vector. The pooling is preferably performed using the max operator.

Let us denote with

$$\text{forward}: \Pi_l(\mathbf{X}) \qquad \text{backward}: \frac{\partial \Pi_i}{\partial \mathbf{X}} \tag{5}$$

the conventional pooling operations where pooling size is fixed at size$(\mathbf{X})/l$, producing l^2 sub-regions.

The forward pass of a Pyramidal Pooling layer, where cat concatenates a set of vectors, can be expressed as

$$y = \forall i \in l \quad . \quad \mathrm{cat}(\Pi_i(\mathbf{X})) \tag{6}$$

and the back-propagation pass can then be expressed by

$$\delta_x = \sum_{i \in l} \frac{\partial \Pi_i}{\partial \mathbf{X}}. \tag{7}$$

Please note that this framework is however not limited to a max-pooling subsampling operation but generalises to any pooling function.

3.2 Multi-scale extraction

A pyramidal feature extraction, while being itself already an improvement over the conventional CNN architecture as it allows to relax the constraint on a fixed input size, extracts features corresponding only to a single scale. We consider a scale 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 pyramidal 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. Attaching a pyramidal pooling before and after a max-pooling operation therefore delivers a multi-scale extraction (Fig. 2).

3.3 Feature encoding layer

The next step in narrowing the gap between the conventional bag-of-features approach and CNN is represented by introducing a feature quantisation layer, the major result of this paper. Descriptors are generally quantised using an overcomplete dictionary resulting in an encoding with only a single or a few non-zero components. This highly non-linear mapping seems to be crucial in obtaining good recognition rates in computer vision benchmarks. It seems then natural to implement an encoding layer also in our framework. If we consider the simplest of the feature quantisation algorithms, k-means with hard assignment, we note that such an encoding algorithm can be approximated by a correlation based measure [3]. This allows us to derive a differentiable approximation of such coding scheme which we name as MLPDict.

[3] $||\mathbf{x} - \mathbf{y}||_2 = \sqrt{\sum_i (x_i - y_i)^2} = \sqrt{\sum_i x_i^2 + \sum_i y_i^2 - 2\sum_i x_i y_i}$, and in the case of $\mathbf{x}$ and $\mathbf{y}$ normalised to have zero mean and unit variance reduces to the correlation between $\mathbf{x}$ and $\mathbf{y}$ as their sum will be almost constant.

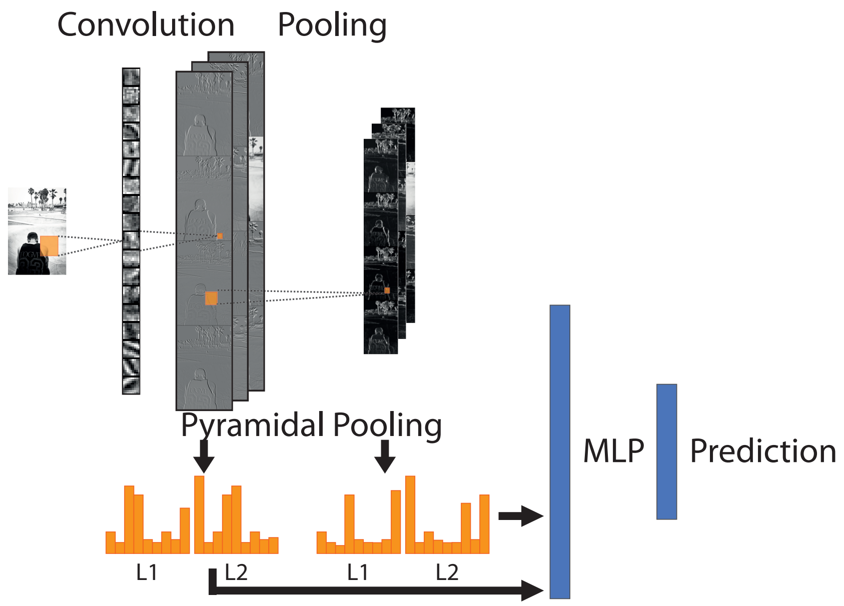

Fig. 2. Schematic representation of a Multi-Scale Pyramidal Pooling Network where histogram-like representations are extracted at two levels and at two scales. The first scale represents the output of the convolutional layer whereas the second scale is given by the output of a pooling (downsampling) layer. The resulting features are concatenated and used as input for a fully connected layer (MLP) which performs classification.

MLPDict Layer: We use a fully connected layer with max-pooling to mimic the behaviour of the k-means coding. The projection of $\mathbf{x}$ with $\mathbf{W}$, the weights of the encoding layer, is the correlation between $\mathbf{x}$ and each column of $\mathbf{W}$. Taking the maximum correlation value is then an approximation of the hard assignment coding scheme. Compared to the bag-of-features approach, $\mathbf{W}$ serves as the dictionary, however an adaptive one which is tuned sample after sample. Moreover our encoding layer together with a non-linear activation function is powerful enough to reduce the dimensionality of the embedding producing much shorter feature vectors which can be processed very quickly in online systems. Performing a pyramidal pooling operation on the result of such an encoding will produce a histogram-like representation.

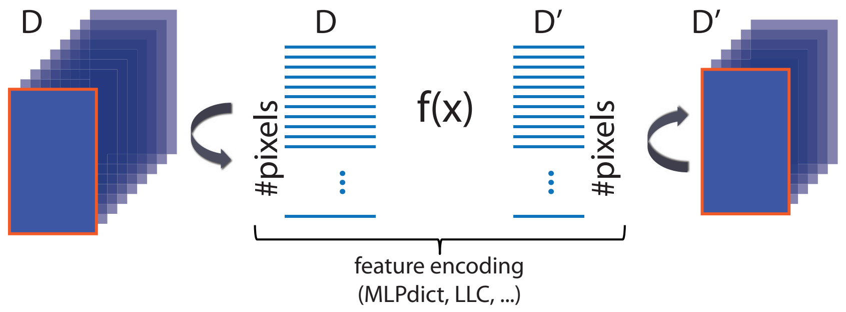

Fig. 3. The MLPdict layer used for feature encoding. Image responses of a network layer are reshaped to produce D dimensional feature vectors, where D represents the number of images in the layer. We consider each pixel as a descriptor and we map them onto another representation of size D' for which only the maxima value per row is preserved.

In Figure 3 a schematic representation of the encoding layer is shown. The hidden representation of a convolutional-based network is composed of D images. We consider each pixel as a D dimensional feature vector (extracted from the densest grid); for the encoding the set of images is reshaped into a matrix with as many rows as pixels in each image of the hidden states and as many columns as images D. Applying a fully connected MLP layer with a weight matrix $\mathbf{W} \in \mathbb{R}^{D \times D'}$ to the reshaped matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ will result in $\mathbf{X}' \in \mathbb{R}^{N \times D'}$ that is reshaped back onto D' images, where N corresponds to the number of pixels in each image. When $D' \ll D$ the layer acts as a feature selection layer which reduces redundancies in the responses and regularises the model when too many filters are used. This is common in image processing, especially in hyper-spectral data processing. When $D' \gg D$ the layer, thanks to the max-pooling operation, operates as a conventional encoding layer such as the one presented in section 2.1. We refer to this quantisation layer as MLPdict and this is what we use in all our experiments.

4 Experiments

In all experiments, unless stated otherwise, we evaluate the average per-class accuracy to establish the classification performance, a more meaningful measure than overall recognition accuracy in case of unevenly distributed datasets. In all experiments no additional preprocessing, such as translation or deformation is used. All nets are trained using stochastic gradient descent with an initial learning rate of 0.001, annealed by a factor of 0.97 at every epoch and a momentum term of 0.9. We usually reached our best result in few training epochs, at most 30, whereas for CNN many more are in general required, especially when distortions and translations are added to the input. We validate our model on several benchmarks and we also show results of a challenging application from the steel industry for which the MSPyrPool framework has been designed for.

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.

MNIST As a reference benchmark to compare our approach with a conventional CNN we take the well studied MNIST [12] benchmark of handwritten characters. CNN excel on this dataset where all digits are of equal size and centred in the middle of a 28×28 grey-scale image. Only for this experiment we use the overall classification accuracy without averaging the per-class performance, to easily compare with other methods. We use a MSPyrPool net with a convolutional layer with 5×5 filters and 100 output maps, a 2×2 max-subsampling layer, 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. A single fully connected layer with softmax is used for classification. Results are shown in Table 1. 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 1. Classification results for the MNIST benchmark. Our MSPyrPool network is compared with other CNN-based approaches which do not use any preprocessing of the input (e.g. no translations nor distortions).

	Test %
CNN LeNet-5 [12]	99.05
CNN + unsup. pre-training [36]	99.40
CNN + unsup. pre-training [20]	99.29
MSPyrPool	99.13

CUReT The Columbia-Utrecht (CUReT) database [37] contains 61 textures, and each texture has 205 images obtained under different viewing and illumination conditions. Results are reported for all 61 textures. For training our architecture only a single image is required as input, just as in previous work [38], with no information (implicit or explicit) about the illumination and viewing conditions.

We use the conventional evaluation protocol [38] but with a different random split of the data. Images are normalised to have zero mean and unit variance to compensate for very different light conditions. We train a CNN with 5 hidden layers: convolutional layer with 11×11 filters and 20 output maps, 5×5 max-subsampling layer, convolutional layer with 9×9 filters and 20 output maps, 5×5 max-subsampling layer, classification layer with softmax activation function. We use *tanh* as activation function for every convolutional layer. We compare the result with a MSPyrPool net composed by: convolutional layer with 11×11 filters and 20 output maps, 5×5 max-subsampling layer, convolutional layer with 9×9 filters and 20 output maps, pyramidal pooling with MLPdict with *tanh* activation and codebook size of 100 at the output of the first and second convolutional layers. Features are pooled using $l = \{1, 2, 4\}$ and $l = \{1, 2, 3\}$ levels to produce a 3500 dimensional vector. An MLP with softmax activation performs the classification stage. Both networks minimise the multi-class cross-entropy loss commonly used in conjunction with a softmax activation. Training is stopped as soon as the training set is fully recognised.

In Table 2 we compare results of our new MSPyrPool network with conventional CNN. The MSPyrPool net generalises much better to the unseen test data and demonstrates the superiority over conventional CNN for the task of texture classification. Furthermore we note that 98.7% recognition rate on all 61 classes is outperforming the bank-of-filter and texton based state-of-the-art 96.4% [38], which share a similar architecture but with pre-wired feature extractors instead. This is a remark that the novel layers are a valuable contribution for the CNN framework and help generalisation.

Table 2. Classification results for the CUReT benchmark. A conventional CNN is compared with our MSPyrPool network.

	Test %
Bank-of-filters + Textons [38]	96.4
CNN	96.5
MSPyrPool	98.7

Caltech101 A further validation of the proposed system is performed on a classical pattern recognition benchmark, Caltech101, where CNN have never successfully been applied. We compare our results with those obtained with a similar system using the bag-of-feature paradigm [29]. We use 30 images per class for training and we test on at most 50 of the remaining images converted

to grey-scale. The feature encoding layer we adopted in this paper is in fact an approximation of the k-means quantisation with hard assignment. We consider a net with and without an encoding layer. Both nets are composed by: convolutional layer with 16×16 filters and 100 output maps, 5×5 max-subsampling layer. 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 that is classified by a fully connected net. 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 that is classified by a fully connected net. 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: convolutional layer with 16×16 filters and 100 maps; 5×5 max-subsampling layer; convolutional layer with 13×13 filters and 100 maps; 5×5 max-subsampling layer; 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 3.

Table 3. Classification results for the Caltech101 benchmark. A MSPyrPool net without an encoding layer (net1), with a linear (net2) and a nonlinear encoding layer (net3) are listed.

	Test %
CNN	25.2
net1	52.8
net2	57.9
net3	55.2
Spatial Pyramid [29]	64.6
LLC [30]	73.4

MSPyrPool nets clearly improve recognition rate compared to a similar sized CNN.[4] Using an encoding layer improves generalisation performance even though the resulting nets have many more free parameters. In this experiment using a non-linear activation degrades generalisation, probably due to the fact that the dictionary size is too big for the non-linear case, and better results might be obtained using much smaller dictionaries in the encoding layer. Nevertheless, the results show that we are able to jointly learn the feature extraction, the quantisation and the classification stages fully online.

Steel-Defects 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 resizing technique without destroying the

[4] Better results are obtained with deeper and bigger CNN, we got 40% with a huge CNN still worse than MSPyrPool.

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 all the images. As a consequence one would need to split the training set and train various networks on subclasses of comparable size [39]. Using a MSPyrPool net avoids such problems because of the input-size independent feature extraction, one of the main contributions of this paper.

For this experiment we use a proprietary dataset of ArcelorMittal from a hot-strip mill containing 34 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 ≈ 20 to a maximum of ≈ 2000 pixels (see Fig. 4 for a subset of defects). Furthermore the dataset is unevenly distributed w.r.t. the number of samples per class. 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. We also limit the maximum size per dimension to 500 pixels to accelerate training by resizing images to have at most 500 pixels for the longest edge.

We compare our system to a set of classifiers trained on commonly used features such as (LBP, HOG, PHOG) using the same training setup described in [39] but with a deeper MLP for classification (500 and 250 units). We use a MSPyrPool with: convolutional layer with 11×11 filters and 20 output maps, 2×2 max-subsampling layer, convolutional layer with 9×9 filters and 20 output maps, classification layer with softmax activation function. *tanh* is used for every convolutional layer. Pyramidal pooling layers with MLPdict, *tanh* activation and codebook size of 100 are attached at the output of the first and second convolutional layers. Features are pooled at $l = \{1, 2, 4\}$ and $l = \{1, 2, 3\}$ levels producing a vector of 3500 dimensions.

The results are summarised in Table 4. 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 of the single features classifiers.

5 Conclusions

We proposed an extension of conventional CNN consisting of three new ingredients: 1) a pyramidal pooling layer that makes the net independent of input image size; 2) a multi-scale feature extraction; 3) an encoding layer emulating standard dictionary-based encoding strategies. We validated the novel architecture on various benchmarks 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, applications that so far have eluded CNN. The proposed extensions open up the possibility to use fully supervised convolutional-based neural nets in many new applications.

Table 4. Classification results for the Steel-defect benchmark where CNN fail. Various classifiers trained on conventional features (LBP, HOG, PHOG) are compared with our MSPyrPool network. Our method outperforms any classifier based on classical computer vision features.

	Test %
LBP	29.1
LBP-HF	48.9
MONO-LBP	64.0
VAR	34.3
HOG	60.5
PHOG	58.9
CNN	-
MSPyrPool	67.9

On the other hand, this work closes the gap between neural network-based models and weak classifiers such as the bag-of-features approach. To the best of our knowledge, ours is the first architecture that learns 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, a new and flexible framework for supervised object classification.

References

1. Lowe, D.: Object recognition from local scale-invariant features. In: The Proceedings of the Seventh IEEE International Conference on Computer Vision. Volume 2. (1999) 1150–1157
2. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In Schmid, C., Soatto, S., Tomasi, C., eds.: International Conference on Computer Vision & Pattern Recognition. Volume 2., INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334 (June 2005) 886–893
3. Bosch, A., Zisserman, A., Munoz, X.: Representing shape with a spatial pyramid kernel. In: Proceedings of the 6th ACM international conference on Image and video retrieval. CIVR '07, New York, NY, USA, ACM (2007) 401–408
4. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). Comput. Vis. Image Underst. **110** (June 2008) 346–359
5. LeCun, Y., Bengio, Y.: word-level training of a handwritten word recognizer based on convolutional neural networks. In IAPR, ed.: Proc. of the International Conference on Pattern Recognition. Volume II., Jerusalem, IEEE (October 1994) 88–92
6. LeCun, Y., Bengio, Y.: Convolutional networks for images, speech, and time-series. In Arbib, M.A., ed.: The Handbook of Brain Theory and Neural Networks, MIT Press (1995)
7. Simard, P., Bottou, L., Haffner, P., LeCun, Y.: Boxlets: a fast convolution algorithm for neural networks and signal processing. In: Advances in Neural Information Processing Systems 11, MIT Press (1999)

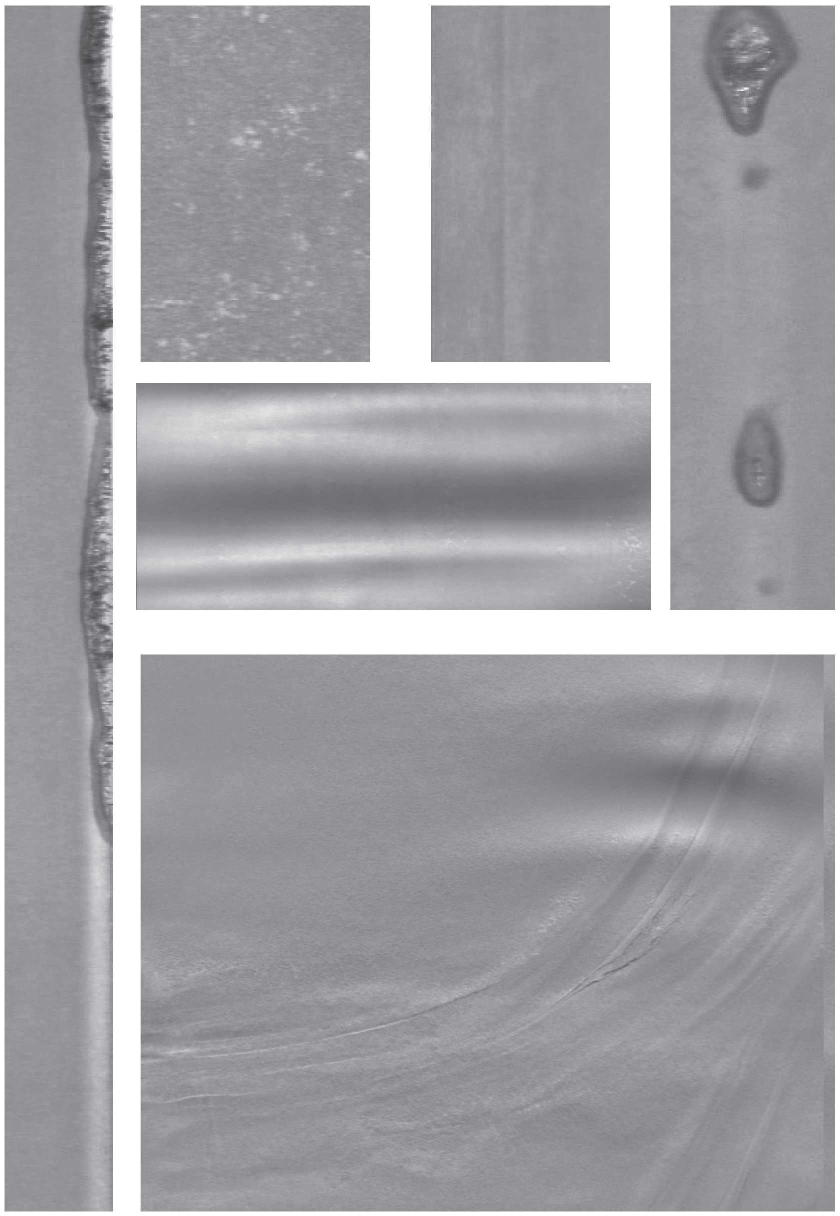

Fig. 4. Subset of images from the Steel-defects benchmark showing the great difference in size among various samples. In this setting is not possible to resize the images to the same size as it will destroy most of them, hence a CNN is not applicable to solve this problem.

8. Lee, H., Grosse, R., Ranganath, R., Ng, A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Proceedings of the 26th International Conference on Machine Learning, New York, New York, USA, ACM Press (2009) 609–616
9. Sivic, J., Zisserman, A.: Video google: a text retrieval approach to object matching in videos. In: Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on. (2003) 1470–1477
10. Csurka, G., Dance, C.R., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: In Workshop on Statistical Learning in Computer Vision, ECCV. (2004) 1–22
11. Fukushima, K.: Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics **36**(4) (1980) 193–202
12. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (November 1998) 2278–2324
13. Behnke, S.: Hierarchical Neural Networks for Image Interpretation. Volume 2766 of Lecture Notes in Computer Science. Springer (2003)
14. Cireşan, D.C., Meier, U., Masci, J., Schmidhuber, J.: Flexible, high performance convolutional neural networks for image classification. In: International Joint Conference on Artificial Intelligence (IJCAI2011). (2011) 1237–1242
15. Hubel, D.H., Wiesel, T.N.: Receptive fields and functional architecture of monkey striate cortex. The Journal of physiology **195**(1) (March 1968) 215–243
16. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Computation (2006)
17. Ranzato, M., Huang, F., Boureau, Y., LeCun, Y.: Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: Proc. Computer Vision and Pattern Recognition Conference (CVPR'07), IEEE Press (2007)
18. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and Composing Robust Features with Denoising Autoencoders. In: Neural Information Processing Systems (NIPS). (2008)
19. Coates, A., Lee, H., Ng, A.: An analysis of single-layer networks in unsupervised feature learning. In: Advances in Neural Information Processing Systems. (2010)
20. Masci, J., Meier, U., Ciresan, D., Schmidhuber, J.: Stacked convolutional auto-encoders for hierarchical feature extraction. In: ICANN'11. 52–59
21. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: Proc. Computer Vision and Pattern Recognition Conference (CVPR'06), IEEE Press (2006)
22. Rigamonti, R., Brown, M.A., Lepetit, V.: Are sparse representations really relevant for image classification? In: CVPR. (2011) 1545–1552
23. Ciresan, D.C., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: Computer Vision and Pattern Recognition. (2012) in press
24. Simard, P., Steinkraus, D., Platt, J.: Best practices for convolutional neural networks applied to visual document analysis. In: Seventh International Conference on Document Analysis and Recognition. (2003) 958–963
25. Ciresan, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Convolutional neural network committees for handwritten character classification. In: ICDAR. (2011) 1250–1254
26. Liu, C.L., Yin, F., Wang, Q.F., Wang, D.H.: ICDAR 2011 chinese handwriting recognition competition. In: 11th International Conference on Document Analysis and Recognition. (2011) 1464–1469

27. Cireşan, D.C., Meier, U., Masci, J., Schmidhuber, J.: A committee of neural networks for traffic sign classification. In: International Joint Conference on Neural Networks (IJCNN2011). (2011) 1918–1921
28. Cireşan, D.C., Meier, U., Masci, J., Gambardella, L.M., Schmidhuber, J.: High-Performance Neural Networks for Visual Object Classification. ArXiv e-prints (February 2011)
29. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: Proc. Computer Vision and Pattern Recognition (VCPR'06), Washington, DC, USA 2169–2178
30. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T.S., Gong, Y.: Locality-constrained linear coding for image classification. In: CVPR, IEEE (2010) 3360–3367
31. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results
32. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09
33. Yang, J., Yu, K., Gong, Y., Huang, T.: Linear spatial pyramid matching using sparse coding for image classification. In: in IEEE Conference on Computer Vision and Pattern Recognition(CVPR. (2009)
34. Jarrett, K., Kavukcuoglu, K., Ranzato, M., LeCun, Y.: What is the best multi-stage architecture for object recognition? In: Proc. International Conference on Computer Vision (ICCV'09), IEEE (2009)
35. Socher, R., Huang, E.H., Pennin, J., Ng, A.Y., Manning, C.D.: Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K., eds.: Advances in Neural Information Processing Systems 24. (2011) 801–809
36. Ranzato, M., Poultney, C., Chopra, S., LeCun, Y.: Efficient learning of sparse representations with an energy-based model. In: Advances in Neural Information Processing Systems (NIPS'06). (2006)
37. Dana, K., Van-Ginneken, B., Nayar, S., Koenderink, J.: Reflectance and Texture of Real World Surfaces. ACM Transactions on Graphics (TOG) **18**(1) (Jan 1999) 1–34
38. Varma, M., Zisserman, A.: A statistical approach to texture classification from single images. Int. J. Comput. Vision **62**(1-2) (April 2005) 61–81
39. Masci, J., Meier, U., Cireşan, D.C., G., F., Schmidhuber, J.: Steel defect classification with max-pooling convolutional neural networks. In: International Joint Conference on Neural Networks (IJCNN2012)