Deep Convolutional Neural Networks for Efficient Vision Based Tunnel Inspection

Konstantinos Makantasis[1], Eftychios Protopapadakis[1] Anastasios Doulamis[2],
Nikolaos Doulamis[2], Constantinos Loupos[3]
[1]Technical University of Crete, Chania, Greece
[2]National Technical University of Athens, Athens, Greece
[3]Institute of Communication and Computer Systems, Athens, Greece
Email: {kmakantasis, eprotopapadakis}@isc.tuc.gr, {adoulam, ndoulam}@cs.ntua.gr, kloupos@iccs.gr

***Abstract*—The inspection, assessment, maintenance and safe operation of the existing civil infrastructure consists one of the major challenges facing engineers today. Such work requires either manual approaches, which are slow and yield subjective results, or automated approaches, which depend upon complex handcrafted features. Yet, for the latter case, it is rarely known in advance which features are important for the problem at hand. In this paper, we propose a fully automated tunnel assessment approach; using the raw input from a single monocular camera we hierarchically construct complex features, exploiting the advantages of deep learning architectures. Obtained features are used to train an appropriate defect detector. In particular, we exploit a Convolutional Neural Network to construct high-level features and as a detector we choose to use a Multi-Layer Perceptron due to its global function approximation properties. Such an approach achieves very fast predictions due to the feed-forward nature of Convolutional Neural Networks and Multi-Layer Perceptrons.**

I. Introduction

One of the greatest challenges facing engineers today is the inspection, assessment, maintenance and safe operation of the existing civil infrastructure, which decay due to various factors. Such situations are common in underground transportation tunnels, a large number of which have been in operation for more than half a century. There are recorded cases of significant failures, resulting in collapses in tunnels in recent years, which highlights the need for better ways to inspect and assess tunnel stability. One should add here that (a) the cost of new tunnel construction is very high and, thus, inspection, assessment and repair of the existing tunnel infrastructure is of utmost importance, (b) the inspection and assessment should be fast in order to minimize tunnel closures or partial closures, and (c) the engineering hours for tunnel inspection and assessment are severely limited.

Visual Inspection (VI) is widely used to determine the safety of concrete structures. VI involves the measurement of defects either manually or in an automatic way. Manual inspection requires significant human effort by involving inspectors walking along the surface of the structure while using only their naked eye. Therefore, a rapid and complete survey cannot be ensured, as it yields subjective results prone to human errors. Automated approaches to the problem of VI consist a challenging alternative to overcome the aforementioned problem.

978-1-4673-8200-7/15/$31.00 ©2015 IEEE

Approaches that utilize automated procedures for VI of concrete infrastructures aim specifically to defects detection by providing objective data to be used for structure evaluation. Towards this direction, such methods exploit image processing and machine learning techniques. Initially, low-level image features are used towards the construction of complex handcrafted features, which in turn are used to train learning models; *i.e.* the detection methods. Automated approaches have been applied in practical settings including roads, bridges, fatigues, and sewer-pipes [1], [2], [3], [4] and [5].

A. Related Work

Most of the approaches that use automated procedures for VI rely on the construction of handcrafted features, which in turn are used by classifiers in order to evaluate concrete infrastructures. Towards this direction Liu *et al.* in [6] utilize image intensity features and the application of Support Vector Machine (SVM) algorithm to detect cracks on tunnel surfaces. Image color properties are also investigated in [7]. Different non-RGB color spaces and various machine learning algorithms, such as Gaussian Mixture Model (GMM), Artificial Neural Network (ANN) and SVM, were evaluated in order to find the optimal combination in terms of detection accuracy.

Abdel-Qader *et al.* in [4] evaluate the effectiveness of four different edge detection techniques on detecting concrete defects. Edge detection algorithms are also utilized by Yu *et al.* in [8]; the Sobel and Laplacian operators are applied on raw image data to extract crack information, which is used to measure cracks through the exploitation of a graph based search algorithm. Mohanty and Wang in [9] are further extending these works by proposing an image mosaic technology for detecting tunnels surface defects which are not identifiable though edge detection.

The work of Koch and Brilakis [10] proposes a pothole detection system in asphalt pavement images. Image segmentation techniques, based on histogram shape-based thresholding and low level texture features, are utilized in order to detect actual potholes. German *et al.* in [11] present a concrete spalling measurement system for post-earthquake safety assessments. Towards this direction, they exploit, at first, a

local entropy-based thresholding algorithm and then template matching techniques and morphological operations.

The exploitation of more sophisticated features, like Histograms of Oriented Gradient (HOG) and shape filtering, has also been proposed. HOG features are utilized in the work of [12], which presents a pattern recognition algorithm to support automated detection and classification of pipe defects using visual cues. Their algorithm employs the HOG algorithm and a SVM to identify pipe defects through a two-step procedure. At first, through image segmentation the algorithm extracts regions of interest that represent candidate defect areas, and then the SVM, which is trained using HOG features, classifies regions of interest as defect or non-defect areas. Shape-based filtering is exploited in the work of [13] to construct features for crack detection and quantification. The constructed features are fed as input to ANN or SVM classifiers in order to discriminate crack from non-crack patterns.

For the evaluation of concrete infrastructures fuzzy/neuro-fuzzy inference has also been utilized. Zhao and Chen in [14] present a fuzzy rule-based inference system for bridge damage diagnosis. A fuzzy partitioning algorithm is implemented to construct the membership functions of the input variables and to induce the fuzzy rules from the numerical data. Similarly, Kawamura and Miyamoto in [15] present a neuro-fuzzy hybrid computing structure for rating deteriorated concrete bridges. Their system is based on ANN utilization and evaluation of bridges takes place on the basis of simple visual inspection. More information about VI in large concrete structures can be found in [16].

B. Our Contribution

Existing approaches follow the conventional paradigm of pattern recognition, which consists of two separate steps; firstly, complex handcrafted features are constructed using the raw input and, secondly, the obtained features are used to train classifiers. However, as it is pointed in [17], there are various defects types, making difficult the feature construction/selection task. In contrast to the conventional paradigm of pattern recognition, deep learning models [18], [19], [20], [21] are a class of machines that can learn a hierarchy of features by building complex, high-level features from low-level ones, thereby automating the process of feature construction for the problem at hand.

In this paper we adopt a deep learning based approach towards the detection of concrete defects in tunnels. In particular, we exploit a Convolutional Neural Network (CNN) to hierarchically construct high-level features from low-level ones, for describing defects, and a Multi-Layer Perceptron (MLP) that carries out the detection task. CNNs have been proposed in 1998 by LeCun et al. [18] and consist a type of deep models, which apply trainable filters and pooling operations on the raw input, resulting in a hierarchy of increasingly complex features. Although, it has been shown that CNNs can achieve superior performance on visual recognition tasks without relying on handcrafted features, due to their nature they produce global image features.

The contribution of our approach can be summarized in three main points; the automated feature extraction, adaptability to the defect type(s), and no need for special set-up for the image acquisition (e.g., specific camera-object distance).

The rest of the paper is organized as follows; section II presents our approach overview. Section III describes the modeling of pixels' visual information, while section IV presents the architecture of deep learning model. Section V shows the experimental results and proposed method's evaluation and, finally, section VI concludes this work.

II. Approach Overview

We consider the detection of concrete defects in tunnels using monocular camera's RGB images. The detection of defects can be seen as an image segmentation problem, which entails, to a great extent, the classification of each one of the pixels in the image into one of two classes; defects class and non-defects class.

Classification requires the description of pixels by a set of highly discriminative features that fuse visual and spatial information. However, the features extraction is depended on the problem at hand. Such drawback can be eliminated following the deep learning paradigm. At first low-level features are extracted over RGB tunnel's surfaces images. These features consist the CNN's input, which fuses them in order to hierarchically construct complex, high-level features, which in turn are fed to a MLP that conducts the classification task. Visual information about a specific pixel is related to its responses to different low-level feature extraction algorithms, while its spatial information is obtained by taking into consideration its neighbors.

After the extraction of low-level image features the raw image has been transformed to a 3D tensor of dimensions $h \times w \times c$, where h and w correspond to the height and width of the image and c to its channels (number of extracted features). In order to be aligned with the specific nature of CNNs (global image features construction), we have to decompose the transformed image into *patches*, each one of which contains visual and spatial information for a specific pixel.

Concretely, in order to classify a pixel p_{xy}, located at (x, y) point on image plane, and successfully fuse visual and spatial information, we use a square patch of size $s \times s$ centered at pixel p_{xy}. If we denote as l_{xy} the class label of the pixel at location (x, y) and as b_{xy} the patch centered at pixel p_{xy}, then, we can form a dataset $D = \{(b_{xy}, l_{xy})\}$ for $x = 1, 2, \cdots, w$ and $y = 1, 2, \cdots, h$. Patch b_{xy} is also 3D tensor, which is divided into c matrices of dimensions $s \times s$ (2D inputs). These matrices are fed as input into the CNN. Then, the CNN hierarchically builds complex, high-level features that encode visual and spatial characteristics of pixel p_{xy}. The output of the CNN is sequentially connected with the MLP. Therefore, obtained features are used as input by the MLP classifier, which is responsible for detecting defects.

III. Visual Information Modeling for Tunnel Inspection

In this section we describe the process for encoding visual information. This process takes place exploiting low-level feature extraction algorithms. Such choice is justified by the fact that low-level features have been successfully used

towards the detection of concrete defects (e.g. [4], [7], [10], [11], [12]). Furthermore, low-level features are calculated over raw image data and thus are less computationally expensive than constructing high-level features. At this point it has to be mentioned that these features are nor used directly for classification purposes. They are used towards the construction of complex, high-level features, which in turn are exploited for the classification task.

Before the low-level feature extraction, the visual information of each one of the image pixels is described by their red, green and blue color components. After the low-level features extraction pixels' visual information is described by pixels' responses to different feature extraction algorithms. In particular, after the feature extraction, each pixel p_{xy} is described by a feature vector $\boldsymbol{f}_{xy} = [f_{1,xy}, \cdots, f_{k,xy}]^T$, where $f_{1,xy}, \cdots, f_{k,xy}$ are scalars correspond to the presence and magnitude of the low-level features detected at location (x, y). In the following we describe, which features are used to form feature vector $\boldsymbol{f}$ and how we address the problem of scale invariance.

A. Edges

In order to successfully exploit images edges, on the one hand the system must be able to detect them in a very accurate way and on the other it must preserve their magnitude. For this reason we combined the Canny and Sobel operators.

Canny operator [22] is a very accurate image edge detector, which outputs zeros and ones for image edges absence and presence respectively. On the other hand, Sobel operator [23], although being less accurate, measures the strength of detected edges by approximating the derivatives of intensity changes along image rows and columns.

Multiplying pixel-wise the output of two operators the system is able to detect edges in a very accurate way, while at the same time it preserves their magnitude. If we denote as C_I and S_I the Canny and Sobel operators for image I then the edges $\mathcal{E}_I$ are defined as

$$\mathcal{E}_I = C_I \cdot S_I \tag{1}$$

The matrix $\mathcal{E}_I$ has the same dimensions with image I and its elements $\mathcal{E}_i(x, y)$ correspond to the magnitude of an image edge at (x, y) location.

B. Frequency

Frequency feature is utilized to emphasize regions of high frequency in the image and at the same time suppress low frequency regions. In order to extract frequency features we utilize the Laplacian operator, which is a second derivative operator; it highlights gray level discontinuities in an image and try to deemphasize regions with slowly varying gray levels. Thus, frequency components for an image I are computed as

$$\mathcal{F}_I = \nabla^2 \cdot I \tag{2}$$

The matrix $\mathcal{F}_I$ has the same dimensions with image I and its elements $\mathcal{F}_i(x, y)$ correspond to the frequency's magnitude at location (x, y) on image plane. Image frequencies are complementary to image edges emphasizing highly structured regions and thus improving detection accuracy.

C. Entropy

Image entropy quantifies the information coded in an image. Images that depict large homogeneous regions, present low entropy, while highly textured images will present high entropy. Image entropy can be interpreted as a statistical measure of randomness, which can be used to characterize the texture of the input image. Therefore, entropy can be used to suppress homogeneous image regions and emphasize potential defect regions, which is expected to be highly textured. Entropy $\mathcal{H}_r$ of a region r of an image is defined as:

$$\mathcal{H}_r = \sum_{j=1}^{k} P_j^{(r)} \cdot \log P_j^{(r)} \tag{3}$$

where $P_j^{(r)}$ is the frequency of intensity j in an image region r. For a grayscale image variable k is equal to 256.

In order to compute entropy for a pixel p_{xy} we apply Eq.(3) on a square window centered at (x, y). Applying this relation on every point of an image I, results to a matrix $\mathcal{H}_I$, which has the same dimensions with image I. The matrix $\mathcal{H}_I$ can be interpreted as a pixel-wise entropy indicator.

D. Texture

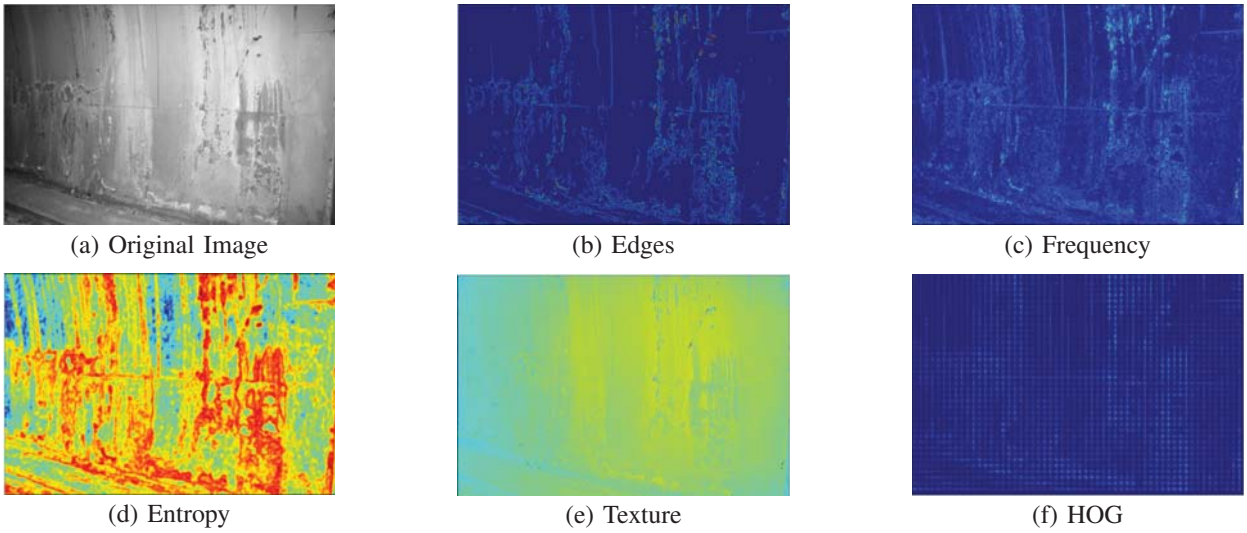
Due to the fact that defect regions is expected to be highly textured, we exploit a separate texture detector capable of identifying different texture forms. Specifically, we utilize Gabor filters. A Gabor filter is characterized by a frequency and an orientation. Frequency and orientation representations of Gabor filters are similar to those of the human visual system, and they have been found to be particularly appropriate for texture representation and discrimination.

In the spatial domain, a 2D Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave. In our implementation we use filters with different frequencies and orientations to extract low-level texture features from the image. Specifically, we use 12 filters with orientations 0°, 30°, 60° and 90° and frequencies 0.0, 0.1 and 0.4.

E. Histogram of Oriented Gradients (HOG)

The HOG is a popular feature descriptor used for the task of object detection. The algorithm for computing HOG consists of five distinct steps. The first step applies an optional global image normalization equalization that is designed to reduce the influence of illumination effects, while the second step computes first order image gradients. These steps capture contour, silhouette and some texture information, while providing further resistance to illumination variations.

The third step aims to produce an encoding that is sensitive to local image content while remaining resistant to small changes in pose or appearance. The adopted method pools gradient orientation information locally in the same way as the SIFT feature [24]. The image plane is divided into small spatial regions, called "cells". For each cell a local 1-D histogram of gradient or edge orientations is accumulated over all the pixels in the cell. This combined cell-level 1-D histogram forms the basic "orientation histogram" representation. Each orientation histogram divides the gradient angle range into a fixed number



(a) Original Image (b) Edges (c) Frequency

(d) Entropy (e) Texture (f) HOG

Fig. 1. Illustration of low-level feature maps. The color scale represents the magnitude of pixels responses to different feature extraction algorithms. Blue and red colors represent low and high magnitude, respectively.

of predetermined bins. The gradient magnitudes of the pixels in the cell are used to vote into the orientation histogram.

The fourth step takes local groups of cells and normalizes their overall responses before passing to next step. Normalization introduces better invariance to illumination, shadowing, and edge contrast. It is performed by accumulating a measure of local histogram "energy" over local groups of cells that are called "blocks". The result is used to normalize each cell in the block. Typically each individual cell is shared between several blocks, but its normalizations are block dependent and thus different. The cell thus appears several times in the final output vector with different normalizations. The normalized block descriptors are referred as HOG descriptors. The final step collects the HOG descriptors from all blocks of a dense overlapping grid of blocks covering the detection window into a combined feature vector.

Following this procedure we construct 16 low-level feature maps for an image. By combining these maps with the raw pixels intensity, feature vector $\boldsymbol{f}$ takes the form of an 1×17 vector containing visual information that characterizes each one of the image pixels. Figure 1 presents extracted feature maps for an image captured at Metsovo tunnel.

F. Scale Invariance

Potential defects usually vary in size. Despite that, most of the feature detectors operate as kernel based methods and thus they prefer objects of a certain size. In our approach a Gaussian image pyramid is exploited to provide scale invariance during feature vectors formation and to take into consideration the relationship between adjacent pixels.

The Gaussian image pyramid is created by successfully low-pass filtering and sub-sampling an image. During the stage of low-pass filtering the Gaussian function can be approximated by a discretized convolution kernel as follows:

$$\boldsymbol{G}_d = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{4}$$

The idea of creating a Gaussian kernel for convolution is to use a 2D Normal distribution as a point-spread function. Since the image is stored as a collection of discrete pixels we need to produce a discrete approximation of the Gaussian function. In theory, the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution kernel, but in practice it is effectively zero more than about three standard deviations from the mean, and so we can truncate the kernel at this point. However, it is not obvious how to pick the values of the kernel to approximate a Gaussian. One could use the value of the Gaussian at the centre of a pixel in the mask, but this is not accurate because the value of the Gaussian varies non-linearly across the pixel. For this reason, the value of the Gaussian is integrated over the whole pixel. Because the integrals are not integers the array is rescaled so that the corners had the value 1. Finally, 256 is the sum of all values in the kernel.

During sub-sampling every even-numbered row and column is removed. If we denote as I^o the original captured image and as I^ϕ the image at pyramid level ϕ then image at pyramid level $\phi + 1$ is computed as:

$$I^{\phi+1}(x, y) = [\boldsymbol{G}_d * I^\phi](2x, 2y) \tag{5}$$

One must combine the various scales together into a single unified and scale-independent feature map, to provide scale-independent feature analysis. To do so, image at level $\phi + 1$, firstly, is upsized twice in each dimension, with the new even rows and columns filled with zeros. Secondly, a convolution is performed with the kernel $\boldsymbol{G}_u$ to approximate the values of the "missing pixels". Because each new pixel has four non

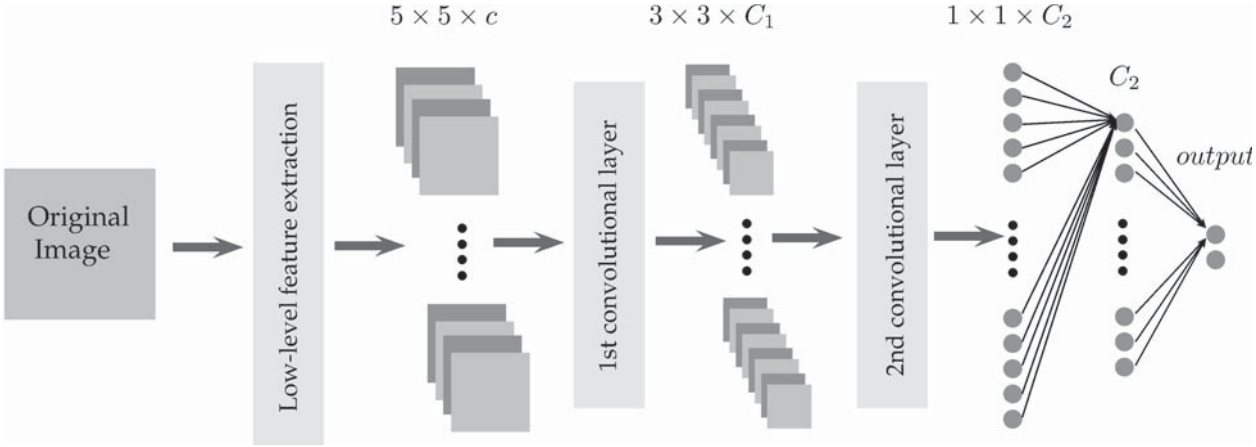


Fig. 2. Deep learning model architecture.

new-created adjacent pixels, $\boldsymbol{G}_u$ is defined as:

$$\boldsymbol{G}_u = 4 \cdot \boldsymbol{G}_d \tag{6}$$

Then, a pixel-wise weighted summation is performed to adjacent images in pyramid so as the unified image at level ϕ, J^{ϕ} is defined as:

$$J^{\phi} = \frac{1}{2} \cdot [I^{\phi} + [\boldsymbol{G}_u * U(I^{\phi+1})]] \tag{7}$$

where U stands for the upsize operation. The final unified image is computed by repeating the above operation, from coarser to finer pyramid image levels. It has to be mentioned that each one of the features, that form the feature vector, is computed independently for each level of pyramid in order to provide scale-independent feature analysis.

IV. Leaning Model Architecture

As it has mentioned before, CNNs apply trainable filters and pooling operations on their input resulting in a hierarchy of increasingly complex features. Convolutional layers consist of a rectangular grid of neurons (filters), each of which takes inputs from rectangular sections of the previous layer. Each convolution layer is followed by a pooling layer that subsamples block-wise the output of the precedent convolutional layer and produces a scalar output for each block.

Formally, if we denote the k-th output of a given convolutional layer as h^k whose filters are determined by the weights W^k and bias b^k then the h^k is obtained as

$$h_{ij}^k = g((W^k * x)_{ij} + b^k) \tag{8}$$

where x stands for the input of the convolutional layer and indices i and j correspond to the location of the input where the filter is applied. Star symbol (*) stands for the convolution operator and $g(\cdot)$ is a non-linear function. Max pooling layers simply take some $k \times k$ region and output the maximum value in that region.

Max pooling layers introduce scale invariance to the constructed features, which is a very important property for object detection/recognition tasks, where scale variability problems may occur. However, for the problem of tunnel defects detection, we involve CNNs to construct features that encode spatio-visual information, which indicates the presence or absence of a defect to a specific pixel. Thus scale invariance, which is addressed through the use of a Gaussian pyramid, does not consist a significant property for our learning model. Due to this fact, we do not involve pooling layers into our learning architecture.

A. Learning Model Parameterization

As mentioned in section III, the input of the CNN are patches of dimensions $s \times s \times c$. After the construction of feature maps, the parameter c corresponds to the dimension of feature vector $\boldsymbol{f}$, in our case it equals 17, while the parameter s determines the number of neighbors of each pixel that will be taken into consideration during classification task.

During experimentation process we set the parameter s to be equal to 5, in order to take into consideration the closest 24 neighbors of each pixel. By increasing the value of s, the number of neighbors that are taken into consideration is increased and thus the computational cost of classification is increased, also. However, setting the parameter s to a value larger than 5, no further improvement on classification accuracy was reported. On the contrary, increasing the value of s over 13, deteriorates classification accuracy.

Having estimate the values of the parameters s and c, we can proceed with the CNN architecture design. The first layer of the proposed CNN is a convolutional layer with $C_1 = 30$ trainable filters of dimensions 3×3. This layer delivers C_1 matrices of dimensions 3×3 (during convolution we do not take into consideration the border of the patch). Due to the fact that we do not employ a max pooling layer, the output of the first convolutional layer is fed to the second convolutional layer.

The second convolutional layer consists of $C_2 = 2 \times C_1$ trainable features. Again, the filters are 3×3 matrices. The second convolutional layer delivers a vector with C_2 elements, which is fed as input to the MLP classifier. The number of MLP hidden units is the same with the dimensionality of its input, *i.e.* the number of hidden units is equal to

Fig. 3. Illustration of various tunnel images during data acquisition process.

60. For training the deep learning architecture the standard back propagation algorithm is employed, in order to learn the optimal model parameters; *i.e.* minimize the negative log-likelihood of the data sets under the model parameterized by MLP weights and filters elements. The learning model architecture is presented in Figure 2.

V. Proposed Methodology Evaluation

The proposed system was developed on a conventional laptop with i7 CPU, 8GB RAM using MatLab software, and Theano library [25] in Python. The proposed CNN is compared against well known techniques in pattern recognition, which rely on the exploitation of handcrafted features. To conduct a fair comparison we use the same features (see section III) for each of these techniques.

A. Data Set Description

All the images were collected in the framework of ROBO-SPECT European FP7 project. They originate from the Metsovo motorway tunnel in Greece, which is a $3,5$km long twin tunnel. Construction of the span north bore was completed in 1994. In a distance of 20m parallel and north to this bore, runs the ventilation tunnel. Although, the last 526m of the ventilation tunnel have been lined with a thick cast in place unreinforced concrete lining, no waterproofing system has been constructed between the temporary support and the final lining. Due to this fact the main tunnel suffered a significant deformation in a Serpentine under an overburden of almost 200m with high ground water inflow.

Image data were captured at this part of the tunnel, using a hand held DSLR camera. During image acquisition no special setup took place, i.e. images are taken from any angle and distance from the tunnel surface. Figure 3 illustrates various tunnel images during data acquisition process. Regions depicting defects, for each one of the captured images, were manually annotated in order to create the ground truth dataset. All algorithm are applied on the raw image data and their performance is evaluated in regard to the ground truth data.

By examining the ground truth dataset we observe that defects span very few areas. This means that the cardinality of the class that represents defects is much smaller than the cardinality of the non-defects class. The unbalanced nature of classes may deteriorate the performance of the classifiers. Due to this fact, we truncate the non-defects class to contain the same number of samples as the class that represents defects. The final dataset that is used for training and testing, is created by concatenating the elements of the two classes.

The final dataset contain, in total, over 100000 samples. We split the dataset into three sets, *i.e.* training, validation and testing data, with a split ration $8:1:1$. That is, we randomly choose 80% of samples as training set, and 10% and 10% for the validation and testing sets respectively.

B. Comparison against other techniques

We use a variety of conventional pattern recognition techniques; all the techniques utilize a variety of parameters (usually user defined). The proposed methods are: **Artificial neural networks** (ANNs), **Support Vector Machines** (SVMs), **k-nearest neighbors** (kNNs) and **Classification trees** (Ctree).

kNNs and Ctrees are rather intuitively methods. The former is a non-parametric method used for classification and clustering. It exploits a similarity metric (usually Euclidean distance) in order to identify the k more similar instances (among the labeled ones) to a specific datum i. The label of datum i is identified by the majority vote of the k closest instances. The latter is a flowchart-like structure in which each internal node represents a rule on feature space, or in other words, a tree shape structure in which every node investigates the values of a specific feature. As we propagate through the tree brunches, by evaluating each of the datum features, we will end to a terminal node, indicating the corresponding class. It is a supervised learning model and thus, tree structure is constructed using the labeled data.

More complex methods, which can capture complex patters are ANNs and SVMs. On the one hand, ANNs are universal function aproximators, which, however, have multiple local minima, *i.e.* solutions, due to their structure. ANNs are composed from multiple hierarchical layers of interconnected nodes/neurons. On the other hand, SVMs delivers a unique solution, since the optimization problem is convex. SVMs select a subset among the labeled data in order to create appropriate separation hyperplane(s) among the classes. SVMs can efficiently perform a non-linear classification using what is called the *"kernel trick"*, implicitly mapping their inputs into high-dimensional feature spaces, where the different classes are linear separable.

Additionally, since we have an abundance of unlabelled pixels in each image semi-supervised learning (SSL) approaches were used. SSL models utilize information of the unlabeled data, under specific assumptions, in order to further improve the classification accuracy. Adopted approaches are **low density separation** (LDS) of [26], as well two graph-based techniques (i.e. **harmonic functions** (Harmonic) [27] and **scalable large graph SSL** (AnchorGraph) [28]).

(a) Original Image (b) Defections

Fig. 4. Output of our proposed method. Defections are denoted with blue color, while non-defect pixels are opaque.

Graph based SSL approaches create a graph over both labeled and unlabeled data. Each datum is a vertex and edges correspond to the similarities among data. Then, from the labeled edges information is propagated to the unlabeled ones using appropriate weight matrices (i.e. the Harmonic approach). Since the scalability is of major concern, $O(n^2)$, there are alternative approaches dealing with the weight graph construction (e.g. AnchorGraph). LDS approach, on the contrary, is a SVMs-like methodology. However, in this case the hyperplane is driven away from dense regions of the data manifold. Density of a region is defined using both labeled and unlabeled data, making the optimization problem non-convex.

C. Performance Metrics

In this paper we have two possible classes; defect or non-defect, named positive (P) and negative (N) class, respectively. We are interested in defect detection, even if we misclassify some non-defects regions as defects. Given the outputs, we form the table of confusion, which is a matrix that reports the number of false positives (FP), false negatives (FN), true positives (TP), and true negatives (TN). Given these values we are able to calculate various performance metrics regarding the defect detection performance. Metrics mathematical formulation is shown in Table I. Metrics of special interest are: Sensitivity (proportional to TP) and miss rate (proportional to FN), which are both strongly connected to defect detection.

Specifically, sensitivity measures the proportion of actual positives which are correctly identified as such, and is complementary to false negative rate. Miss rate, or otherwise false negative rate, is the proportion of actual positives that are being tested and found to belong to the negative class, i.e., the conditional probability to assign a sample to the negative class given that it belongs to the positive class.

TABLE I. METRICS FOR QUATNITATIVE PERFORMANCE EVALUATION.

Metric	Formula
Sensitivity - (TPR)	TPR = TP / P
Specificity - (SPC)	SPC = TN / N.
Precision - (PPV)	PPV = TP / (TP + FP)
Neg. predictive value - (NPV)	NPV = TN / (TN + FN)
False pos. rate - (FPR)	FPR = FP / N
False discovery rate - (FDR)	FDR = 1 - PPV
Miss Rate - (FNR)	FNR = FN / P
Accuracy - (ACC)	ACC = (TP + TN) / (P + N)
F1 score - (F1)	F1 = 2 TP / (2 TP + FP + FN)

D. Performance Evaluation

A quantitative evaluation, in terms of the performance metrics described in subsection V-C, of the proposed method compared to conventional pattern recognition methods is shown in Table II. All learning models were trained and tested on the same datasets. The metric scores presented in Table II correspond to classifiers performance on test set.

The proposed methodology outperforms all competitive techniques. It is worth mentioning that it is capable of detecting almost 90% of defects, while at the same time it presents the lowest value for the false negative rate (FNR), without sacrificing the classification accuracy for the non-defect class (as shown by the ACC, FDR and F1 score).

That suggests a well adapted classifier, able to distinguish between the two classes and showcases the deep learning approach potential for accurate tunnel inspection. Finally, due to the feed forward nature of CNN and MLP the proposed methodology is able to evaluate 1000 pixels in 0.18 seconds. Furthermore, the implementation can be easily parallelized on GPU (Theano's library special design supports GPU implementation), to achieve even faster predictions. Figure 4 visually presents classification results of the proposed method, over a specific image from Metsovo motorway tunnel.

VI. CONCLUSIONS

In this paper, we point the suitability of deep learning architectures for the tunnel defect inspection problem. In contrast to the conventional pattern recognition paradigm, which relies on the construction of handcrafted features, the deep learning approaches allow the hierarchical construction of complex, high-level features. This way the most appropriate features for the specific problem at hand are automatically constructed, minimizing the feature construction effort. We compared our method against well-known supervised and semi-supervised learning approaches. We evaluated the classification accuracy of each one of the classifiers using a variety of performance metrics. As experimental results show, the deep learning approach outperforms all other techniques, showcasing our method's potential for accurate tunnel inspection. Finally, due to the feed forward nature of CNNs and MLPs, our method, on the one hand can achieve very fast prediction and, on the other, can scale to large datasets.

ACKNOWLEDGMENT

The research leading to these results has received funding from the EC FP7 project ROBO-SPECT (Contract N.611145). Authors wish to thank all partners within the ROBO-SPECT consortium.

REFERENCES

[1] J. Pynn, A. Wright, and R. Lodge, "Automatic identification of cracks in road surfaces," in *Image Processing and Its Applications, 1999. Seventh International Conference on (Conf. Publ. No. 465)*, vol. 2, 1999, pp. 671–675 vol.2.

[2] Y.-S. Kim and C. T. Haas, "A model for automation of infrastructure maintenance using representational forms," *Automation in Construction*, vol. 10, no. 1, pp. 57–68, Nov. 2000.

[3] P.-C. Tung, Y.-R. Hwang, and M.-C. Wu, "The development of a mobile manipulator imaging system for bridge crack inspection," *Automation in Construction*, vol. 11, no. 6, pp. 717–729, Oct. 2002.

TABLE II. COMPARISON OF STATE-OF-THE-ART TECHNIQUES FOR VARIOUS PERFORMANCE METRICS. THESE METRICS REPRESENT THE ABILITY TO DISTINGUISH DEFECT AREAS OVER A GIVEN IMAGE.

		Quantitative Performance Metrics (Test Set)								
		TPR	SPC	PPV	NPV	FPR	FDR	FNR	ACC	F1 score
Learning Models	CNN	**0,890**	**0,883**	**0,883**	**0,890**	**0,117**	**0,117**	**0,110**	**0,886**	**0,886**
	Ctree	0,721	0,591	0,751	0,553	0,409	0,249	0,279	0,673	0,736
	kNN	0,845	0,575	0,773	0,685	0,425	0,227	0,155	0,746	0,807
	FFNN	0,854	0,554	0,766	0,689	0,446	0,234	0,146	0,743	0,808
	linSVMs	0,833	0,514	0,746	0,643	0,486	0,254	0,167	0,716	0,787
	polySVMs	0,877	0,036	0,609	0,146	0,964	0,391	0,123	0,567	0,719
	rbfSVMs	0,864	0,470	0,736	0,669	0,530	0,264	0,136	0,719	0,795
	Harmonic	0,668	0,534	0,710	0,485	0,466	0,290	0,332	0,619	0,689
	LDS	0,875	0,524	0,759	0,710	0,476	0,241	0,125	0,746	0,813
	AnchorGraph	0,890	0,530	0,764	0,737	0,470	0,236	0,110	0,757	0,822

[4] I. Abdel-Qader, O. Abudayyeh, and M. E. Kelly, "Analysis of edge-detection techniques for crack identification in bridges," *Journal of Computing in Civil Engineering*, vol. 17, no. 4, pp. 255–263, 2003.

[5] S. K. Sinha and P. W. Fieguth, "Automated detection of cracks in buried concrete pipe images," *Automation in Construction*, vol. 15, no. 1, pp. 58–72, 2006.

[6] Z. Liu, S. A. Suandi, T. Ohashi, and T. Ejima, "Tunnel crack detection and classification system based on image processing," in *Electronic Imaging 2002*. International Society for Optics and Photonics, 2002, pp. 145–152.

[7] H. Son, C. Kim, and C. Kim, "Automated color modelbased concrete detection in construction-site images by using machine learning algorithms," *Journal of Computing in Civil Engineering*, vol. 26, no. 3, pp. 421–433, 2012.

[8] S. Yu, J. Jang, and C. Han, "Auto inspection system using a mobile robot for detecting concrete cracks in a tunnel," *Journal of Computation in Civil Engineering*, vol. 17, no. 4, pp. 255–263, 2006.

[9] A. Mohanty and T. T. Wang, "Image mosaicking of a section of a tunnel lining and the detection of cracks through the frequency histogram of connected elements concept," vol. 8335, 2012, pp. 83 351P–83 351P–9.

[10] C. Koch and I. Brilakis, "Pothole detection in asphalt pavement images," *Advanced Engineering Informatics*, vol. 25, no. 3, pp. 507–515, Aug. 2011.

[11] S. German, I. Brilakis, and R. DesRoches, "Rapid entropy-based detection and properties measurement of concrete spalling with machine vision for post-earthquake safety assessments," *Advanced Engineering Informatics*, vol. 26, no. 4, pp. 846–858, Oct. 2012.

[12] M. R. Halfawy and J. Hengmeechai, "Automated defect detection in sewer closed circuit television images using histograms of oriented gradients and support vector machine," *Automation in Construction*, vol. 38, pp. 1–13, Mar. 2014.

[13] M. R. Jahanshahi, S. F. Masri, C. W. Padgett, and G. S. Sukhatme, "An innovative methodology for detection and quantification of cracks through incorporation of depth perception," *Machine Vision and Applications*, vol. 24, no. 2, pp. 227–241, Feb. 2013.

[14] Z. Zhao and C. Chen, "A fuzzy system for concrete bridge damage diagnosis," *Computers & structures*, vol. 80, no. 7, pp. 629–641, 2002.

[15] K. Kawamura and A. Miyamoto, "Condition state evaluation of existing reinforced concrete bridges using neuro-fuzzy hybrid system," *Computers & structures*, vol. 81, no. 18, pp. 1931–1940, 2003.

[16] C. Koch, S. Paal, A. Rashidi, Z. Zhu, M. Knig, and I. Brilakis, "Achievements and challenges in machine vision-based inspection of large concrete structures," *Advances in Structural Engineering*, vol. 17, no. 3, pp. 303–318, Mar. 2014.

[17] M. Halfawy and J. Hengmeechai, "Integrated vision-based system for automated defect detection in sewer closed circuit television inspection videos," *Journal of Computing in Civil Engineering*, vol. 29, no. 1, p. 04014024, 2015.

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[19] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006. [Online]. Available: http://www.sciencemag.org/content/313/5786/504

[20] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[21] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems 19*, B. Schlkopf, J. C. Platt, and T. Hoffman, Eds. MIT Press, 2007, pp. 153–160. [Online]. Available: http://papers.nips.cc/paper/3048-greedy-layer-wise-training-of-deep-networks.pdf

[22] W. McIlhagga, "The Canny Edge Detector Revisited," *Int J Comput Vis*, vol. 91, no. 3, pp. 251–261, Oct. 2010. [Online]. Available: http://link.springer.com/article/10.1007/s11263-010-0392-0

[23] I. Yasri, N. Hamid, and V. Yap, "Performance analysis of FPGA based Sobel edge detection operator," in *International Conference on Electronic Design, 2008. ICED 2008*, Dec. 2008, pp. 1–4.

[24] D. Lowe, "Object recognition from local scale-invariant features," in *The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999*, vol. 2, 1999, pp. 1150–1157 vol.2.

[25] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[26] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation," Sep. 2004.

[27] X. Zhu, J. Lafferty, and Z. Ghahramani, "Combining active learning and semi-supervised learning using gaussian fields and harmonic functions," in *ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, 2003, pp. 58–65.

[28] W. Liu, J. He, and S.-F. Chang, "Large graph construction for scalable semi-supervised learning," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 679–686.