Deep CNNs for Object Detection using Passive Millimeter Sensors

Santiago López-Tapia, Rafael Molina, and Nicolás Pérez de la Blanca

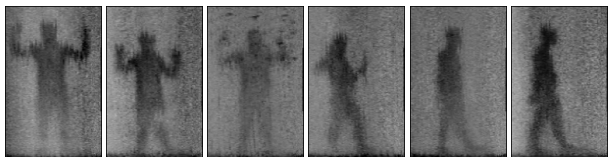
***Abstract*—Passive Millimeter Wave Images (PMMWIs) can be used to detect and localize objects concealed under clothing. Unfortunately, the quality of the acquired images and the unknown position, shape, and size of the hidden objects render these tasks challenging. In this paper we discuss a deep learning approach to this detection/localization problem. The effect of the non stationary acquisition noise on different architectures is analyzed and discussed. A comparison with shallow architectures is also presented. The achieved detection accuracy defines a new state of the art in object detection on PMMWIs. The low computational training and testing costs of the solution allow its use in real time applications.**

***Index Terms*—Millimeter wave imaging, deep learning, object detection, security, classification.**

I. Introduction

Millimeter images are obtained by sensors capturing electromagnetic radiation in the 0.001-0.01 m wavelength range. Two types of sensors can be distinguished: active, which direct the waves to the subject and then collect and interpret the reflected energy; and passive, which create images using the radiation emitted by objects, in general, and human bodies in particular.

In contrast to alternatives like backscatter X-ray, passive millimeter systems are safe, see the interesting discussion in [1]. Furthermore, they fully respect the privacy of their users. Unfortunately, millimeter sensors, and consequently their images, suffer from, among others, the following problems: low signal to noise ratio, low resolution, which can be increased by increasing the sampling rate but at the cost of decreasing the signal to noise ratio, and in-homogeneous signal intensity.



(a) (b) (c) (d) (e) (f)

Fig. 1: PMMWI examples. Hidden objects correspond to whiter areas within the body. Unfortunately, not all whiter areas correspond to hidden objects.

Research funded by the Spanish Ministry of Economy and Competitiveness (MINECO) under projects TIN2013-43880-R and DPI2016-77869-C2-2-R, FEDER Funds, and project TIC-1692 (Junta de Andalucía). We would like to acknowledge the use of the real PMMWIs provided by Wavecamm (http://wavecamm.com)

Santiago López-Tapia, Rafael Molina, and Nicolás Pérez de la Blanca are with the the Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain (emails: {sltapia,rms,nicolas}@decsai.ugr.es)

PMMWIs (see Fig. 1) are currently being utilized as theft and threat detection systems [2] in places like airports and warehouses. Since April 2009, many PMMWI systems have been installed in large USA and European airports. These detection systems have to deal with the unknown position, shape, size and transmission properties of the hidden objects. Unfortunately, their false positive ratios are very high. Systems based on PMMWIs should be able to detect concealed objects, incur in a very low number of false positive detections, and work in real time. It is worth mentioning here that the scientific community has, so far, shown little interest on the challenging tasks involved in threat detection using PMMWIs. This is likely because of the absence of large databases of PMMWIs to work with.

Sensor modelling and image processing techniques have been used on PMMWIs. In [3], the main concepts related to millimeter images are introduced, see also [4]. In [5], trends in millimeter wave imaging technologies are examined, focusing mainly on applications and technical parameter variations for security surveillance and nondestructive inspections. Image processing techniques have also been utilized on PMMWIs. The use of compressive sensing and superresolution techniques on these images is explored in [6], [7]. Denoising, deconvolution, and enhancement techniques have also been applied to this kind of images, see for instance [8]–[11]. In this paper, we develop an image classification approach to detect and localize objects in a context where the image noise is non stationary and very high.

II. Related Works

K-means is used in [12] to segment PMMWIs into three regions: background, body and threats. Unfortunately, the method detects unconnected areas. To solve this problem, the authors use Active Shape Models (ASM) inside the body. However, this approach does not guarantee an adequate segmentation. In [13], Gaussian mixture models are used to characterize background, body, and threat regions and segment the image. Although the reported results are better than those in [12], this method also produces an unconnected body segmentation. In [14], the authors apply noise elimination and then image segmentation using Local Binary Fitting (LBF). The authors use two algorithms for noise removal: Non Local Means (NLM) and Iterative Steering Kernel Regression (ISKR). Although its detection rate is around 90%, its computing time makes it impractical for real-time applications. Furthermore, its performance decreases significantly when used on noisy or low quality images. In [15], a fast two-step algorithm, based

on denoising and mathematical morphology, was proposed. On noisy or low contrast images it achieves an acceptable detection rate but at the cost of a high false positive detection rate. In [16], a method to detect and recognize threats in outdoor PMMWIs is proposed. The threat detection is carried out through global and local segmentation: at each level (global and local), a Gaussian mixture model, whose parameters are initialized using vector quantization, is used and optimized through expectation maximization. For a different initialization approach see [17]. Finally, a Bayesian decision rule decides which cluster each pixel belongs to. The recognition process of the threat type consists of upscaling, principal component analysis (PCA), size normalization, extraction of a geometric-based feature vector composed of shape descriptors, and a decision rule, where the class is decided by minimum Euclidean distance between normalized feature vectors. The method was able to detect threats, but it was tested exclusively on a small set of images and with only two types of threats. In [18], the same segmentation process as in [16] was used, the difference being the initialization of the Gaussian mixture model using k-means clustering. Shape features from the detected object (area, perimeter, major and minor axes of the basic rectangle, rectangularity, compactness, and eccentricity) are extracted and compared to the true features. The method shows good accuracy. Notice that all the above approaches aim at segmenting the concealed objects, furthermore they were evaluated on a small set of images. In [19], [20], we introduced the UGR-PM^{2}WI database described in appendix B, a new and comprehensive dataset of PMMWIs. A comparative study of the performance of different shallow classifiers on this database was presented. Although the proposed classifiers, and in particular Random Forest, outperform previous approaches, it was shown that the image noise (see Fig. 1) has a strong influence on this kind of classifiers, making very difficult to do better than a 68% average detection score.

All the above approaches can be grouped according to the spatial context used in the feature extraction process: a) local spatial context, used in object detection/localization tasks and b) full image context, mainly used in image segmentation tasks. In what follows, these approaches will be denoted Detection using a Local Approach (DLA) and Segmentation using a Global Approach (SGA), respectively.

In the Deep Learning (DL) methodology we are going to propose, deep SGA and DLA approaches will be analyzed and compared. Both approaches will be cast under the same framework, the one provided by Convolutional Neural Network (CNN) architectures used for supervised classification problems. The difference between both approaches will be characterized by the class of functions computed by the deep CNN architectures.

The rest of the paper is organized as follows. Section III provides a short introduction to the used CNN architectures. In section IV, we describe how the threat detection problem can be tackled using deep classifiers. The detection and training procedures are also discussed in this section. In section V, we describe the experimental set-up and show the obtained results. Also, a comparison with the best shallow architectures is also presented here. In section VI, we analyze the performance of the different learning architectures, identifying the most relevant factors providing high detection score. Conclusions are drawn in section VII. Appendix A contains a description of the denoising technique applied to the observed image, the importance of its use is described in section VI. Appendix B contains a complete description of the dataset used in the experiments. [1].

III. Deep architectures

Nowadays, Feature Learning Models (FLM) compete with advantage against the standard classification methodology where a feature extraction process is carried out to obtain the information to feed a classifier. FLMs learn the best set of features by minimizing the training error. Deep CNN models have shown the importance of using deep architectures as a mechanism to disentangle independent features coded in the data through a very complex non-linear mapping of the sample data. CNNs are currently the most successful DL models for high level image related tasks like classification, segmentation, detection, parsing, etc. CNNs define the state of the art performance in many of them [21]–[27].

Although our threat detection problem could be naively approached by the direct application of well-known image processing techniques, the problem is much more challenging. As shown in this paper, the presence of non-stationary noise in the images renders extremely difficult the detection task. The use of function classes as complex as those defined by DL architectures is required.

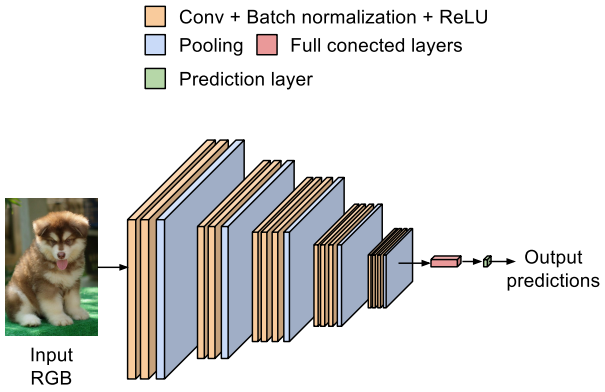


Fig. 2: Example of a CNN architecture.

A. Used architectures

The LeNeT architecture introduced by Lecun [28] can be considered as the base model from which more sophisticated CNN architectures have been designed. A CNN model is a feedforward neural net defined as a deep composition of functions or layers, where each layer computes a transformation from an accepted list of transformations: convolution, pooling, non-linear activation, full connection, regularization and normalization. The design of a specific model consists of two phases. Firstly, we have to fix the architecture, that is,

[1]The dataset is available at http://decsai.ugr.es/pi/pmmwi/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2774927, IEEE Transactions on Circuits and Systems for Video Technology

to define the number, the order, and the characteristics of the layers. Secondly, we must learn the free parameters associated to the convolution and fully connected layers. In the last layer, a final vector is obtained which is used to feed a classifier. We use a softmax classifier, as it provides a posterior probability estimation for each class. A simple CNN architecture is shown in Fig. 2.

For our problem, we have analyzed five different CNN architectures. Three of them, with increasing level of complexity, for the DLA approach and two for the SGA approach. The learning protocols for DLA and SGA are different. DLA models are fed with image patches while SGA ones utilize full images.

Let us start by describing the three architectures which utilize image patches. The simplest model is an adaptation of the LeNeT [29] architecture. We use the same type and number of layers as the original, 2 convolutional layers, 2 pooling layers and 2 fully connected layers, but the number of filters in the first and second convolutional layers has been changed from {20,50} to {32,64} filters, respectively. Also the original non-linear sigmoidal function is changed to a ReLU [30] activation function, $f(x) = max(0, x)$. Batch normalization [31] has also been introduced before each ReLU transformation to decorrelate the outputs of the convolutional layers and, finally, regularization with dropout [32] has been added to the last fully connected layers. All these changes adjust the original architecture [29] to our threat detection problem and produce better classification figures. Fig. 3 shows a summary of the this low deep CNN architecture (LD-CNN). This model has 12 layers and 1,621,598 free parameters. The central column of the figure describes the order and functionality of each layer (top-down).

	Input	Block output
Block 1	Conv. 5x5 32 ReLU	32x39x39
Block 2	Max-pooling 2x2	64x20x20
	Conv. 5x5 64 ReLU	
Block 3	Max-pooling 2x2	64x10x10
	Conv. 5x5 64 ReLU	
	Dropout 50%	
Block 4	Full connected 500 ReLU	500
	Dropout 50%	
Block 5	Full connected 2 Softmax	2

Fig. 3: Description of the LD-CNN architecture. The total number of parameters is 1,621,598. By Dropout x% we mean that a x% of the pre-activation units are assigned to zero.

A very recent architecture named All-CNN-C [33] implements pooling and fully connected layers with convolutional layers. This architecture allows us to train models similar to LD-CNNs but with larger number of layers. The main advantage of this architecture is that it only depends on convolution operations which are very well adapted to GPU hardware. A max-pooling layer can be seen as a convolution with kernel size equal to the kernel size of the pooling followed by a non-linear transformation and a downsampling operation. This means that each max-pooling layer can be replaced by three convolution layers. In addition, as a bonus, the pooling parameters can be included as parameters to optimize. A fully connected layer can also be replaced by convolutional layers with a 1×1 kernel and the same number of kernels as output units has the fully connected layer. Therefore, all fully connected layers are replaced by convolution layers.

We adopt this All-CNN-C [33] architecture to design our medium deep CNN model (MD-CNN) shown in Fig. 4. Now we have an architecture with almost twice the number of layers the LD-CNN has, but in terms of effective transformations it is only slightly deeper than LD-CNN. Two convolutional layers implement transformations equivalent to max-pooling and the layers in block 5 implement transformations equivalent to a fully connected layer. The total number of free parameters is 1,371,458. Notice that, although MD-CNN is more complex and powerful than LD-CNN, the number of parameters to be estimated is lower.

	Input	Block output
Block 1	Dropout 20%	96x39x39
	2xConv. 3x3 96 ReLU	
Block 2	Conv. 3x3 96 ReLU /2	96x20x20
	Dropout 50%	
Block 3	2xConv. 3x3 192 ReLU	192x20x20
Block 4	Conv. 3x3 192 ReLU /2	192x10x10
	Dropout 50%	
	Conv. 3x3 192 ReLU	
	Conv. 1x1 192 ReLU	
Block 5	Conv. 1x1 2 ReLU	2x10x10
Block 6	Global average pooling	2
	Softmax	

Fig. 4: Description of the MD-CNN architecture. The total number of parameters is 1,371,458. See dropout x% meaning in Fig. 3. In this model a convolution layer with stride 2 is equivalent to a pooling and sub-sampling stage.

Training very deep networks is usually difficult since the optimization procedure frequently gets trapped in very poor local minima. The Residual Network approach [34] overcomes this problem by fitting on each new layer a residual function f of the form $out = f(input) + input$ instead of using the standard function defined by $out = f(input)$. Fig. 5 shows a standard residual unit as described in [34]. In order to perform the sum operation $(f(input) + input)$ when input and output do not have the same size, we project the input into the output space using another convolutional layer (the one in green in Fig. 5) before the element-wise sum, X is the number of output feature maps and Y the number of input feature maps of the unit. Adapting the architecture proposed in [34], a very deep architecture with 20 layers, most of them residual, has been created (DR-CNN). Fig. 6 shows the proposed architecture. The three final layers are fully connected. In this case, a more adequate activation function PReLU [35] ($f(x) = max(-\epsilon x, x),\ 0 < \epsilon << 1$), which

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2774927, IEEE Transactions on Circuits and Systems for Video Technology

allows activation for negative values, is used. The main effect of the PReLU function is to improve the behaviour of the gradient estimation in the bottom layers. The number of free parameters is 2,847,138.

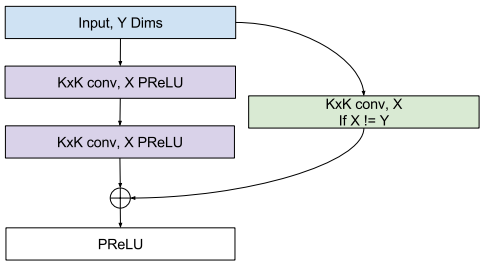


Fig. 5: Residual unit used in Deep-Residual-CNN. The input is transformed throughout two convolutional layers with PReLUs to learn the residual. See in text how the sum operation is performed.

	Input	Block output
Block 1	Conv. 7x7 32 PReLU /2	32x20x20
Block 2	Max-pooling 2x2	32x10x10
	2xResidual 3x3 32	
Block 3	Residual 3x3 64 /2	64x5x5
	Residual 3x3 64	
Block 4	Residual 3x3 128 /2	128x3x3
	Residual 3x3 128	
Block 5	2xResidual 3x3 256	256x3x3
Block 6	Global average pooling	256
Block 7	Full connected PReLU 128	128
	Dropout 20%	
Block 8	Full connected PReLU 64	64
	Dropout 20%	
Block 9	Full connected 2 Softmax	2

Fig. 6: Description of the Deep-Residual-CNN (DR-CNN) architecture. The residual units correspond to the standard residual unit described in [34], see Fig. 5. The total number of parameters is 2,847,138. See dropout x% meaning in Fig. 3.

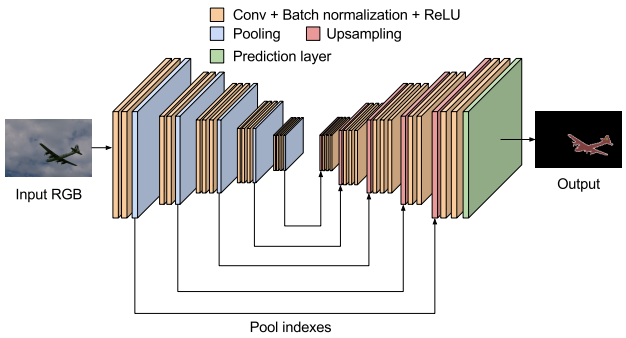


Fig. 7: Segnet architecture as described in [36].

Once we have described the three CNN architectures used for the DLA approach, we study now the two SGA methods. Our problem can be considered a segmentation task where the regions of interest are defined by the grouping of high-level gray pixels using context conditions. When the main goal is to learn features encoding the location of regions of interest, the use of CNNs has proven to be a successful approach [36]. A typical segmentation architecture is shown in Fig. 7. The feature learning process takes place on the left side of the architecture (decreasing in size layers). The smallest layer on the left represents the learned features which encode threat locations. Since the loss function used to learn the architecture must be evaluated on all image pixels, the layers on the right side (increasing in size layers) decode the estimated label, in our case, a probability of containing a threat from the learned features. In summary, the SGA approach uses architectures that take as input the whole image I and output for each pixel a probability distribution P on the label set.

This approach has been used recently in image segmentation tasks [36] on natural images. The original model, Segnet, has 29,458,886 parameters where the encoding layers are designed by the combination of convolution, ReLU and max-pooling layers. In the decoding part (right side of Fig. 7), the model carries out the spatial propagation of the high level features using an inverse process named max-unpooling. The combination of convolutional, max-unpooling, and ReLU layers defines the spatial propagation of the high level features found by the encoder until the size of the input image has been matched. The max-unpooling mapping only assigns values to the units whose indexes were local maxima in the corresponding max-pooling operation of the encoding part.

The Segnet architecture was designed to solve a much more complex (in terms of class number) problem than ours. We adapt this approach to our task by designing two new architectures but keeping the encoder-decoder approach. The first model, shown in Fig. 8, has 193,827 parameters. We refer to it as SCNN. The probability distribution for each pixel is obtained using a sigmoidal activation on the output of the last convolutional layer to map the values to the $[0, 1]$ interval.

It is important to note here that autoencoders and, more recently, Generative Adversarial Networks are generative deep networks architectures also using an encoding-decoding approach [37]. The main goal of these architectures is to learn features and parameters to represent the generative distribution of the samples. Although this learning problem is clearly much more difficult and complex than ours, the use of deep generative models on our data is an area worth to explore.

The second model makes use of multiscale information. Multiscale architectures have been used to improve the performance of CNNs [38]. The use of multiscale features for PMMWI classification with shallow classifiers [19], [20] has also provided better classification results. This insight has led us to modify the SCNN in order to incorporate information at three different scales (s, $s/2$ and $s/4$). We process each scale to produce k feature maps using an SCNN on each scale. We then combine the $3 * k$ feature maps with 2 convolutional layers after resizing the output of the lower scales. Finally, for each pixel the probabilities are computed from regularized

convolutional layer and a sigmoidal activation. To eliminate possible redundancies between scales, we apply spatial dropout (instead of dropping a single unit, we drop an entire feature map) before the scale combination. We call this model SCNN-MS, it has 355,515 parameters. Its architecture can be seen in Fig. 9.

	Input	Block output
Block 1	Conv. 11x11 16 ReLU	16x195x125
Block 2	Max-pooling 2x2	32x97x62
	Conv. 7x7 32 ReLU	
Block 3	Max-pooling 2x2	64x48x31
	Conv. 5x5 64 ReLU	
Block 4	Max-pooling 2x2	32x24x15
	Conv. 3x3 32 ReLU	
Block 5	Conv. 3x3 64 ReLU	64x48x31
	Max-unpooling 2x2	
Block 6	Conv. 5x5 32 ReLU	32x97x62
	Max-unpooling 2x2	
Block 7	Conv. 7x7 16 ReLU	16x195x125
	Max-unpooling 2x2	
Block 8	Conv. Sigmoid 11x11 1	195x125

Fig. 8: Description of the SCNN architecture. The total number of parameters is 193,827. The max-unpooling layer is the same used in the segnet architecture [36].

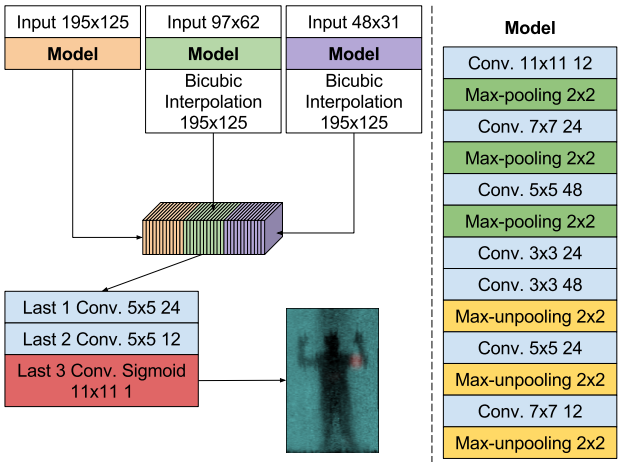


Fig. 9: Description of the SCNN-MS architecture. The total number of parameters is 355,515. The red area contains pixels with high probability of being threat.

IV. The approach

Given an image, the goal is to classify it as Class +1 (positive) when it contains at least one concealed object or Class 0 (negative) otherwise. We also need to localize where the concealed object is on the human body.

Let $\mathcal{D} = \{\mathcal{I}_i, i = 1, \cdots, N_I\}$ be the image dataset. Our goal is to learn from $\mathcal{D}$ a function $f_{\mathcal{D}}(\mathcal{I}_i) = \mathcal{X}_i$, with the lowest generalization error. This function assigns to each image the set of pixels, $\mathcal{X}_i$, where threats are located or an empty set if no threat is present. This problem, as we have already indicated, can be cast as either an object detection/localization problem using local image patches or a segmentation one when the whole image is used, both are approached here using machine learning techniques. We define the function $f_{\mathcal{D}}$ as the composition of two functions; one detecting potential threat regions and the other deciding which of the regions should be considered a real threat. The full dataset $\mathcal{D}$ is used to learn the function, $f_{\mathcal{D}}$, and its generalization error is estimated by cross validation.

A. The training set $\mathcal{T}$

We start by splitting the set of images $\mathcal{D}$ in five folds to estimate the generalization error of $f_{\mathcal{D}}$ by five fold cross-validation. Four folds are used for training and one for validation. Each fold has the same number of images with one, two o more threats. Due to the small size of the threat regions and the possibility of having several threats in the same image, a possible approach to learn the detection function is to use image patches. A patch is a rectangular image piece centered on a pixel. By $\mathbf{x}_j^P$ we denote the patch $\mathbf{x}^P$ centered on pixel j. Let P_I be a mapping which selects a subset of patches $\mathcal{P}_I^i$ from the image $\mathcal{I}_i$, that is $P_I(\mathcal{I}_i) = \mathcal{P}_I^i$. Each one of the patches is binary labelled, $(+1)$ or (-1), depending on whether it contains a threat or not. The patch dataset associated to each image fold is defined by $\mathcal{T}_j = \cup_{k=1}^{N_I} \mathcal{P}_I^k \; j = 1, \cdots, 5$. On each cross-validation iteration, the union of four of the patch datasets is used to learn the corresponding detection function $f_{\mathcal{P}_j} : \mathcal{T}_j \rightarrow [0, 1]$ and the other is used for validation. A binary classifier is built by splitting the $f_{\mathcal{P}_j}$ range using a calibrated threshold.

Taking into account the object sizes described in appendix B, three different patch scales can be identified as representatives of the object sizes: 39×39, 19×19, and 9×9 pixels, respectively. From each image we only extract the 39×39 patches that are fully included inside the image. The use of the largest size patches forces the model to learn threats in different spatial contexts. Later on we show that the joint use of the three scales is relevant when the whole image is considered as input.

The full set of training patches extracted from four image folds is huge (see appendix B). Then, to reduce the computational cost of the training process, we extract per image one patch every 2×2 pixels, obtaining a total of $3,476$ patches per image. In total we have $11,502,084$ patches of size 39×39. We have found no loss in classification performance by this training dataset reduction. The 39×39 patches covering a full object are positively labelled (1), the rest are negatively labelled (-1). Patches that partially overlap an object are not included in the training dataset. We have $372,494$ positive instances and $9,351,130$ negative ones. Since the number of negatives is around 25 times the number of positives and taking into account that most of the negative patches are very similar, we have used for training a subset of them. Specifically, we keep only one negative sample every 2×2 block, so we retain $1/4$ of the negative patches we have. The final number of negative

samples, $2,337,782$, is then around six times the number of positive ones.

When the complete image is considered as input, the learning process is simply a standard five fold cross-validation.

B. Learning $f_{\mathcal{P}_j}$

To label the 39×39 image patches as threat or non-threat, we train the [LD,MD,HD]-CNN architectures. To deal with the imbalanced training set, a sampling strategy is used. We partition the set of negative patches and learn an ensemble of classifiers. Let $\mathcal{T}_P, \mathcal{T}_N$ be the subsets of positive and negative labelled patches, respectively, and $\mathcal{T} = \mathcal{T}_P \cup \mathcal{T}_N$. Let us denote by n_P and n_N their cardinality, being $n_P << n_N$ with $n_C = n_N/n_P$. Let $\mathcal{T}_N = \cup_{k=1}^{n_C} \mathcal{T}_N^k$ be a random decomposition of $\mathcal{T}_N$ in n_C disjoint subsets. We solve n_C learning problems, $f_{\mathcal{P}}^k,\ k = 1, \cdots, n_C$, associated to the training sets defined by $\{\mathcal{T}_N^k \cup \mathcal{T}_P\}$, respectively. We repeat the same procedure t times obtaining an ensemble of $t \times n_C$ functions $f_{\mathcal{P}}^k$ which use the positive and subsets of the negative patches. Then, the patch classifier is defined using a threshold on the range of the weighted average of the $f_{\mathcal{P}}^k$, this is $f_{\mathcal{P}} = \sum_k w_k f_{\mathcal{P}}^k$ with weights $w_k > 0$ and $\sum_k w_k = 1$, alternatively as a weighted majority voting on the set of classifier provided by $\{f_{\mathcal{P}}^k\}$. In our experiments we have used the majority voting rule to decide whether a patch is a potential threat, and its detection score is measured by $\frac{1}{t \times n_C} \sum_k f_{\mathcal{P}}^k(\mathbf{x}_P)$. In the experiments we use $n_C = 6$ and $t = 1$. This means that the $\mathcal{T}_N$ subset is split in six random partitions and each partition together with the $\mathcal{T}_P$ subset is used to fit a new model.

The segmentation approaches represented by SCNN and SCNN-MS architectures use full images as input. In these cases, the label is defined by a binary matrix of the same size as the input one, with ones in pixels covered by the threats and zeros in the rest.

C. Initialization and loss function

In all cases, the initialization of the weights of the layers is done by random sampling from a Gaussian distribution with $\text{mean} = 0$ and $\text{std} = \sqrt{\frac{2}{\#\text{input\_unit}}}$ [35] , where $\#\text{input\_unit}$ represents the number of input units to each layer.

The fitting loss function used by all our models is the negative log likelihood given by

$$L(\mathbf{x}, \mathbf{y}; \theta) = -\sum_j y_j \log p(c_j|\mathbf{x}), \tag{1}$$

where $\mathbf{y}$ represents the target distribution and $p(c_j|\mathbf{x})$ denotes, for all the models, the probability of class j defined by the softmax function of the output layer.

D. Hyperparameters

The LD-CNN, MD-CNN and DR-CNN models were trained using Stochastic Gradient Descend (SGD) with minibatches of size 64, variable learning rate, and 0.9 momentum. The variable learning rate policy follows a triangular scheme [39] that consists of varying the learning rate between a minimum and a maximum value following a triangular pattern with the training iterations. The triangular learning rate parameters range from 0.01 to 0.04 for the LD-CNN and MD-CNN models and from 0.01 to 0.03 for the DR-CNN model. All these models have been trained using a total of 4 epochs, two of them increasing uniformly the learning rate value with the iterations from the minimum until reaching the maximum and two more going back, that is, we decrease the learning rate similarly from the maximum to the minimum value. We apply batch normalization before each nonlinear transformation. Also we perform local brightness and contrast normalization on each image [40], that is, we remove the mean and divide by the standard deviation of its elements.

SCNN and SCNN-MS were also trained using SGD and 0.9 momentum, additionally we had to use the Nadam [41] learning rate estimation for the method to converge. In these cases, minibatches of size 16 were used. We fix Nadam hyper-parameters to the standard values except the learning rate, which was set to 0.02. In SCNN-MS the learning rate was set to 0.002 after the first 16 epochs. SCNN was trained for 16 epochs, while SCNN-MS used 20 epochs. In all cases, batch normalization before each nonlinear transformation and before the first layer was also applied.

E. Regularization

All the models were regularized during training using weight decay with a 0.005 fixed value. Dropout regularization was also applied to the LD-CNN, MD-CNN and DR-CNN models. For the LD-CNN model a 50% dropout layer before every full connected layer was used. As suggested in [33], for the MD-CNN model a 20% dropout layer on the input data and 50% after every subsampling phase was introduced (see Fig. 4). In the DR-CNN models we only apply a 20% dropout to the output of the layers 15 and 18 (see Fig. 6).

To improve the convergence of the fitting process for our SCNN and SCNN-MS models, we corrupt the input image with random Gaussian noise of 0 mean and 0.05 standard deviation. Also random Gaussian noise of 0 mean and 0.0005 standard deviation was applied to the output of each layer (except for the last one). Spatial dropout with 50% intensity was applied to SCNN-MS after concatenating the three scales feature maps.

F. Detection and validation

The ROC curve associated to each trained model is used to calibrate its detection threshold, $\text{thr}_\text{M} \in [0, 1]$. $\text{Pr}_\text{M}(S) \geq \text{thr}_\text{M}$, M$\in$ {LD-CNN, MD-CNN, DR-CNN, SCNN, SCNN-MS}, will declare a potential threat in S. Here S represents a patch or the whole image.

The LD-CNN, MD-CNN, and DR-CNN models provide probabilities on overlapping patches. To fuse this information a non-maximum suppression step is carried out. We reject a patch $\mathbf{x}_j^P$ if it has an intersection-over-union overlap larger than a learned threshold with a higher scoring patch $\mathbf{x}_i^P$. That is, if $\text{Pr}_\text{M}(\mathbf{x}_i^P) > \text{Pr}_\text{M}(\mathbf{x}_j^P) \geq \text{thr}_\text{M}$, then the patch $\mathbf{x}_j^P$ is removed (see Fig. 10).

Let us denote by $\mathcal{H}_\mathcal{S}$ and $\mathcal{P}_\mathcal{S}$ the support region (set of pixels) in the image associated to a hidden object and a patch respectively. In the experiments, we consider that a patch $\mathbf{x}^P$

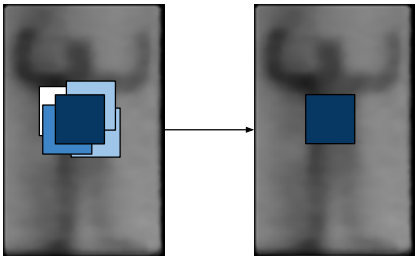


Fig. 10: The non-maxima suppression process. Blue rectangles indicate regions whose probabilities are above the minimum probability threshold (darker regions correspond to higher probability). Since these regions overlap more than the overlapping threshold, we only maintain the one with highest probability (in the image, the darkest rectangle).

is correctly classified as positive when $f_{\mathcal{P}}(\mathbf{x}^P) > \text{thr}_\text{M}$ and $\mathcal{H}_\mathcal{S} \cap \mathcal{P}_\mathcal{S}(\mathbf{x}^P) > 0.5 * \text{size}(\mathcal{H}_\mathcal{S})$. A hidden object in the image is correctly detected when there is at least one patch which satisfies both conditions. Finally, an image is correctly classified as positive when, at least, a hidden object has been detected.

For SGA methods, the above approach is applied to the full image. We consider in each image the same patch structure used for patch classification.

V. Experimental setup

Our experimental setup has been designed to answer the following questions: a) what is the influence of the image noise on the models?; b) how relevant is the depth to the detection?; c) is it important to use multiscale information?; d) should deep learning models be used to solve this task?.

We ran experiments with all the architectures in two scenarios, raw and filtered images (see appendix A). This allows us to evaluate when smoothing the image helps the detection process. AUC, TP, and FP figures of merit are calculated as the average of the five values obtained from each cross-validation set. Means and standard deviations are included. AUC represents the area under the ROC curve when the validation images are classified as containing or not containing hidden objects. TP indicates the percentage of detected hidden objects. FP represents the average number of false positives per image. The threshold and overlap parameters used when non-maxima suppression was applied are also reported (see subsection IV-F).

We performed all our experiments using Caffe [42] and Keras [43] with Theano [44]. We used an Intel Xeon E5-2630 v3 at 2.40GHz with 8 cores and 256GB of Ram and 1 GPU Nvidia Titan X (Pascal) with 12GB of Ram.

VI. Discussion

Table I shows the obtained classification results with CNN architectures. It also shows how the best shallow architecture: Random Forest concatenating three scales of Haar features (RF-HAAR-MS) [19], [20] performs. We have also evaluated the role of regularization on the learning process for the SCNN and SCNN-MS models (adding noise and spatial dropout). The obtained results with no regularization, not included here, show an important decrease in their figures of merit when regularization is not used.

A comparison among the different classifiers trained from image patches, [LD,MD,DR]-CNN and RF-HAAR-MS, shows that an important improvement is obtained when very deep models are used (DR-CNN). The LD-CNN model has a much lower score than the shallow RF-HAAR-MS. This indicates that a simple architecture with a low number of layers is unable to deal with the image noise to extract good features. The MD-CNN performs slightly better than RF-HAAR-MS pointing out that increasing the number of layers introduces robustness against non-stationary noise. The DR-CNN model obtains a better score, with an AUC value of almost 79%. This shows that deep enough models are able to extract robust and informative features on noisy scenarios. Architectures using preprocessed images are noted *-pre in Table I.

TABLE I: CNN results. These models are defined by a committee of classifiers using raw and preprocesed (pre) images respectively. For some architectures we also show the result using preprocessed images, only the best performing ones are shown. See the text for additional explanation on the columns.

Model	AUC $\times 10^2$	TP$\times 10^2$	FP	Thr	Overlap
LD-CNN	55.3 ± 2.2	92 ± 3.7	7.1 ± 7.0	0.7	0.4
MD-CNN	70.9 ± 1.7	92 ± 1.9	6.3 ± 5.9	0.6	0.3
DR-CNN	78.8 ± 1.8	95 ± 2.2	4.2 ± 4.1	0.5	0.3
DR-CNN-pre	69.3 ± 2.1	95 ± 1.1	6.2 ± 6.0	0.65	0.3
SCNN	86.2 ± 1.4	100	0.02 ± 10^{-5}	0.65	0.5
SCNN-pre	83.4 ± 2.7	100	0.03 ± 0.04	0.65	0.5
SCNN-MS	$\mathbf{87.4 \pm 0.5}$	**100**	$\mathbf{0.006 \pm 10^{-6}}$	**0.65**	**0.5**
SCNN-MS-pre	82.1 ± 4.2	100	0.1 ± 0.12	0.65	0.5
RF-HAAR-MS	75.3 ± 1.6	94 ± 0.9	4.0 ± 3.8	0.7	0.3

The best performing CNN methods are the ones fed with full images instead of patches. They can deal better with the spatial non-stationarity image noise and take advantage of analyzing the whole image at once. SCNN performs very well, reaching a 86% AUC, a 100% rate of positive detection and average of 0.02 false positives per image (FP). In comparison with the patch based architectures, these figures represent an enormous improvement. When multiscale information is considered in SCNN-MS, an additional improvement is achieved, reaching 87.4% AUC while at the same time decreasing significantly the FP rate by nearly one order of magnitude. As it can be observed, the best score, in bold, shows an over 12 point improvement over the score obtained by the best shallow architecture, RF-HAAR-MS, with the added benefit of having eliminated the feature extraction process. In terms of detection errors (TP and FP), these deep learning architectures also show an overwhelming improvement over the shallow ones. These findings clearly point out the negative influence of the image noise on the shallow architectures and their difficulty to remove the noise in the classification process. On the contrary, the deep learning models show robustness to image noise when the number of layers increases. It is also interesting to note that image processing does not help any CNN model. This could be explained by noticing that any image preprocessing technique introduces new spatial dependencies on the image values, making a much harder task for the model to extract

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2774927, IEEE Transactions on Circuits and Systems for Video Technology

uncorrelated features. In other words, the preprocessing step masks the signal and makes more difficult for the models to extract uncorrelated features. Although the reported figures correspond to applying one particular filter, see Appendix A, we also tried other filters with similar results.

Figure 11 shows some examples of the most relevant activation maps of the SCNN-MS model "last 2 Conv" layer (see Fig. 9). The first column contains the input images, columns Map [1-3] show the most relevant activation maps contributing to the classification layer. This layer before the sigmoidal function is applied is shown in the fifth column (see Before-σ column). The other nine activation maps do not contain much internal structure. They do not contribute much to the detection process. However, if they are removed the performance of the method decreases slightly. The last column shows the input image with the detected threat in green. The first two rows (True Positive) show examples where our approach detects a true threat. It can be seen that the areas with high activation in the column Before-σ correlate with lack of activation in the maps. In rows three and four (True Negative) the behavior is similar, but the intensity of the final activation is smaller and after the sigmoidal transformation it is eliminated. The last two rows (False Positive) show two examples where our model fails. In this case no threat is present but the model identifies a false positive. The discussion above suggests that the filters look for areas of high contrast in the images. Since our model detects all the threats in the database (see Table I), no false negative examples are shown.

We finally report training and test times. Obviously, the training time depends on the number of free parameters to be estimated and the number of convolutions to be carried out. Table II depicts the training and test computing times for all the CNN architectures. It is interesting to note the high influence of the architecture design on the training and test figures. Clearly those models trained using image patches require higher training times due to a much larger training set, additionally the layer design has a strong influence, see the [LD,MD,HD]-CNN models. The MD-CNN model needs much more time than the other models due to the higher number of convolutions associated to the higher number of filters. On the contrary, the SCNN model is much faster in both training and test times. This property together with its high efficiency makes it a very good candidate for real-time applications.

TABLE II: CNN training and test times (h hours, m minutes and s seconds). We show the one fold mean training time, the test time over all the images, and the mean time per image.

Model	Training one fold	Total test	Per image test
LD-CNN	1h and 32m	47.42m	0.86s
MD-CNN	14h and 27m	746.82m	13.54s
DR-CNN	9h and 50m	73.90m	1.34s
SCNN	**2m and 24s**	**48s**	**0.015s**
SCNN-MS	5m and 52s	2m and 24s	0.044s
RF-HAAR-MS	19h and 12m	12m and 20s	0.22s

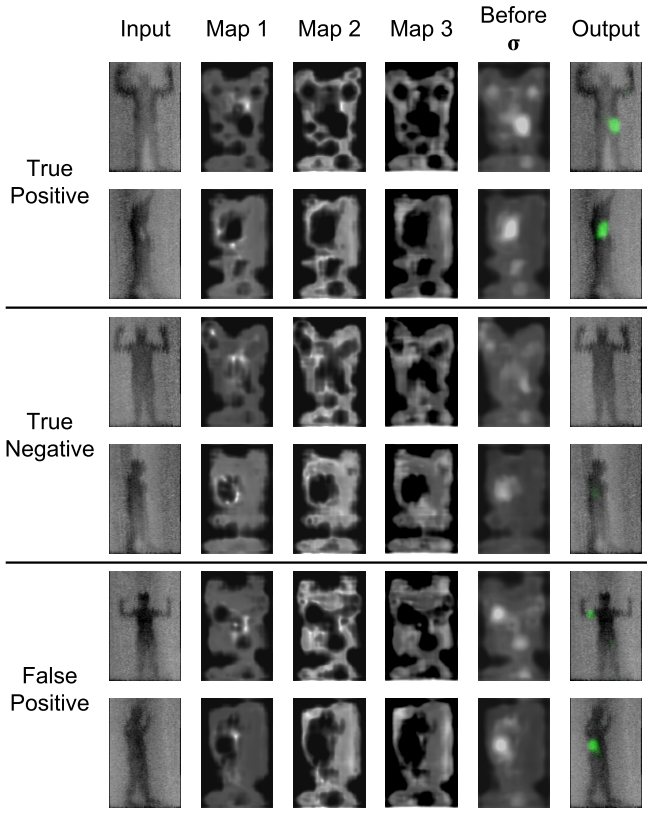


Fig. 11: Examples of the most relevant activation maps of the SCNN-MS "Last 2 Conv" layer. The first column contains the input images. Columns 2-4 display the activations of the most relevant filters. The fifth column shows the activations of the last convolutional layer just before the sigmoidal (σ) activation. The last column shows in green the predicted probabilities for each pixel over the input images (higher probabilities are associated to greener intensities). Notice that low responses in the feature maps correspond to high responses before the sigmoidal and the predicted highest probability zones (see text). Activation layers have been rescaled for better visualization.

VII. Conclusions

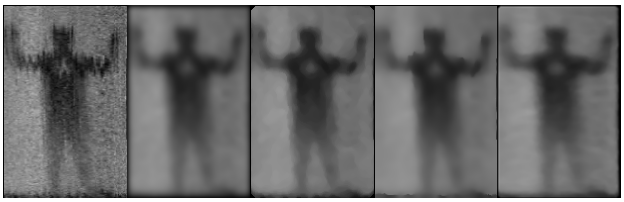
In this paper we have analyzed the use of CNN-DL classifiers to detect and localize concealed objects in PMMWIs approach. We have shown that to address the non-stationary noise present in the images, deep architectures should be used. They show an overwhelming improvement over the best performing shallow ones. We have also found that the two most important factors for this improvement have been the depth of the architecture and feeding the models with the whole image. The combined use of information from different scales improves the detection score and also greatly reduces the classification errors. The regularization process applied to our model has shown to be crucial in order to find a model with a high detection score (AUC, TP) and a low error rate (FP).

The obtained results allow us to conclude that our DL-CNN architectures define a new state of the art for the task of detecting and localizing concealed objects in PMMWIs. In addition, we have found that deep architectures are a very good

tool to deal with signals immersed in non-stationary noise. Our experimental set-up uses the new UGR-PM^{2}WI dataset, to the best of our knowledge, the currently most comprehensive available dataset for this task.

APPENDIX

A. Image Enhancement



(a) (b) (c) (d) (e)

Fig. 12: Results of applying different image filtering (Best view in high resolution screen), see text for details.

A natural question to consider is whether image filtering (smoothing, contrast enhancement, etc.) helps the detection process. Fig. 12 shows examples of applying mean (b), median (c), and bilateral (d) filters to the observed image, Fig. 12(a).

We observed that a better smoothing criterion is to assign to each pixel an estimation of the most frequent value in its neighbourhood. That is, for each pixel i, let $\mathbf{y}_i^{\mathrm{B}}$ denote a block around y_i in the observed image $\mathbf{y}$, $z_i(1), \ldots, z_i(K)$ denote K independent samples with replacement from $\mathbf{y}_i^{\mathrm{B}}$. We then redefine

$$q_i = \frac{\sum_{k=1}^{K} z_i(k)}{K}, \tag{2}$$

assign $\mathbf{y} = \mathbf{q}$ and repeat the process L times. The final processed image is

$$x_i = \mathrm{median}(\mathbf{y}_i^B) \tag{3}$$

We have found experimentally that $B = 5 \times 5$ and $L = 5$ produce good processed images, see Fig. 12(e). Notice that some contrast improvement can be observed. See the experimental section to analyse how image processing influences the performance of the proposed classifiers.

B. UGR-PM^{2}WI Dataset

A comprehensive dataset of PMMWIs has been created. It consists of 3309, 125×195 images of 33 different people. Hidden objects are in the range of 35×39 to 10×10 pixels, which in our images corresponds roughly to $2752.39cm^2$ and $201.64cm^2$, respectively. Smaller hidden objects are not considered to be relevant.

We took pictures of different people (with different complexions and heights) wearing 12 different objects in 10 body locations: forearm, chest, stomach, thigh, ankle (front), waist (side), armpit (side), arm, ankle (lateral), thigh (lateral), and 2 images without any objects. Images of people wearing simultaneously two objects in different locations were also taken. In summary, the dataset consists of 463 pictures of people with no object, 2144 containing one object and 702 containing 2 objects.

Threats were simulated by bags containing objects/substances with different millimeter wave responses: a cutter, 325g of gel, a 200g clay bar, a simulated gun, 200g of sugar, 200g of frozen peas, 150ml of cologne, 160g of gel, a bag with metal pieces, 200g of flour, a 50cl water bottle, and a 250ml hydrogen peroxide bottle (see Fig. 13). Notice that different object sizes were used.

Fig. 14 shows PMMWIs of subjects with simulated threats on different locations. It is important to note that although in the experiments objects are visible and not hidden under clothing, this is irrelevant to PMMW sensors.

Fig. 13: Simulated threats in the dataset: a cutter (1), 325g of gel (2), a 200g clay bar (3), a simulated gun (4), 200g of sugar (5), 200g of frozen peas (6), 150ml of cologne (7), 160g of gel (8) , a bag with metal pieces (9), 200g of flour (10), a 50cl water bottle (11), and a 250ml hydrogen peroxide bottle (12).

The visual images were taken at the same time as the millimeter ones. Objects are marked on the visual images by the smallest bounding boxes containing them. To transfer object bounding boxes from visible images to PMMWI ones, a homography, estimated from the visible image plane to the PMMWI one using a calibration pattern, was applied. These bounding boxes will later be used to assess the performance of the classifiers.

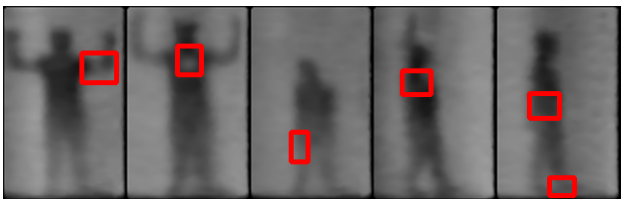


Fig. 14: Examples of PMMWIs with (hidden) objects. Red boxes indicate object locations. Objects to be detected correspond to whiter areas on the body.

REFERENCES

[1] J. Accardo and M. A. Chaudhry, "Radiation exposure and privacy concerns surrounding full-body scanners in

airports," *Journal of Radiation Research and Applied Sciences*, vol. 7, no. 2, pp. 198 – 200, 2014.

[2] N. E. Alexander, C. Callejero Andrés, and R. Gonzalo, "Multispectral mm-wave imaging: materials and images," *Proceedings of Society of Photo-Optical Instrumentation Engineers (SPIE)*, vol. 6948, pp. 694803–694803–10, 2008.

[3] L. Yujiri, M. Shoucri, and P. Moffa, "Passive millimeter wave imaging," *IEEE Microwave Magazine*, vol. 4, no. 3, pp. 39–50, 2003.

[4] L. Yujiri, "Passive millimeter wave imaging," in *2006 IEEE MTT-S International Microwave Symposium Digest*, pp. 98–101, 2006.

[5] S. Oka, H. Togo, N. Kukutsu, and T. Nagatsuma, "Latest Trends In Millimeter-wave Imaging Technology," *Progress in Electromagnetics Research Letters*, vol. 1, pp. 197–204, 2008.

[6] S. D. Babacan, M. Luessi, L. Spinoulas, A. K. Katsaggelos, N. Gopalsami, T. Elmer, R. Ahern, S. Liao, and A. Raptis, "Compressive passive millimeter-wave imaging," in *2011 18th IEEE International Conference on Image Processing*, pp. 2705–2708, 2011.

[7] W. Saafin, S. Villena, M. Vega, R. Molina, and A. K. Katsaggelos, "Compressive sensing super resolution from multiple observations with application to passive millimeter wave images," *Digital Signal Processing*, vol. 50, pp. 180–190, 2016.

[8] B. Han, J. Xiong, L. Li, J. Yang, and Z. Wang, "Research on millimeter-wave image denoising method based on contourlet and compressed sensing," in *Signal Processing Systems (ICSPS), 2010 2nd International Conference on*, vol. 2, pp. V2–471–V2–475, 2010.

[9] J. Mateos, A. López, M. Vega, R. Molina, and A. Katsaggelos, "Multiframe blind deconvolution of passive millimeter wave images using variational Dirichlet blur kernel estimation," in *IEEE International Conference on Image Processing*, pp. 2678–2682, 2016.

[10] J. Yang, J. Wang, and L. Li, "A new algorithm for passive millimeter-wave image enhancement," in *Signal Processing Systems (ICSPS), 2010 2nd International Conference on*, vol. 3, pp. V3–507–V3–511, 2010.

[11] W. Yu, X. Chen, S. Dong, and W. Shao, "Study on image enhancement algorithm applied to passive millimeter-wave imaging based on wavelet transformation," in *Electrical and Control Engineering (ICECE), 2011 International Conference on*, pp. 856–859, 2011.

[12] C. Haworth, B. Gonzalez, M. Tomsin, R. Appleby, P. Coward, A. Harvey, K. Lebart, Y. Petillot, and E. Trucco, "Image analysis for object detection in millimetre-wave images," in *Passive Millimetre-wave and Terahertz Imaginh and Technology*, vol. 5619, pp. 117–128, 2004.

[13] C. Haworth, Y. Petillot, and E. Trucco, "Image processing techniques for metallic object detection with millimetre-wave images," *Pattern Recognition Letters*, vol. 27, no. 15, pp. 1843 – 1851, 2006.

[14] O. Martínez, L. Ferraz, X. Binefa, I. Gómez, and C. Dorronsoro, "Concealed object detection and segmentation over millimetric waves images," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pp. 31–37, June 2010.

[15] I. Gómez Maqueda, N. Pérez de la Blanca, R. Molina, and A. Katsaggelos, "Fast millimetre wave threat detection algorithm," in *European Signal Processing Conference (EUSIPCO 2015)*, pp. 599–603, Nice (France), September 2015.

[16] S. Yeom, D.-S. Lee, Y. Jang, M.-K. Lee, and S.-W. Jung, "Real-time concealed-object detection and recognition with passive millimeter wave imaging," *Optics Express*, vol. 20, pp. 9371–9381, Apr 2012.

[17] W. Yu, X. Chen, and L. Wu, "Segmentation of concealed objects in passive millimeter-wave images based on the gaussian mixture model," *Journal of Infrared, Millimeter, and Terahertz Waves*, vol. 36, no. 4, pp. 400–421, 2015.

[18] S. Yeom, D.-S. Lee, and J.-Y. Son, "Shape feature analysis of concealed objects with passive millimeter wave imaging," *Progress in Electromagnetics Research Letters*, vol. 57, pp. 131–137, 2015.

[19] S. López-Tapia, R. Molina, and N. Pérez de la Blanca, "Detection and localization of objects in passive millimeter wave images," in *24th European Signal Processing Conference*, pp. 2101–2105, Aug 2016.

[20] S. López Tapia, R. Molina, and N. Pérez de la Blanca, "Detection and localization of concealed objects in passive millimeter images," *Engineering Applications of Artificial Intelligence*, vol. 67, pp. 81–90, 2018.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[22] C.-Y. Lee, P. W. Gallagher, and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree.," *CoRR*, vol. abs/1509.08985, 2015.

[23] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *CoRR*, vol. abs/1511.07289, 2015.

[24] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, vol. 28, pp. 1058–1066, JMLR Workshop and Conference Proceedings, May 2013.

[25] B. Graham, "Fractional max-pooling," *CoRR*, vol. abs/1412.6071, 2014.

[26] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014.

[27] F. Liu, G. Lin, and C. Shen, "CRF learning with CNN features for image segmentation," *Pattern Recognition*, vol. 48, pp. 2983–2992, Oct. 2015.

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, November

1998.

[30] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, Omnipress, 2010.

[31] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)* (D. Blei and F. Bach, eds.), pp. 448–456, JMLR Workshop and Conference Proceedings, 2015.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[33] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, "Striving for simplicity: The all convolutional net," *CoRR*, vol. abs/1412.6806, 2014.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

[36] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *arXiv preprint arXiv:1511.00561*, 2015.

[37] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[39] T. Schaul, S. Zhang, and Y. LeCun, "No More Pesky Learning Rates," *CoRR*, vol. abs/1206.1106, June 2012.

[40] A. Coates, H. Lee, and A. Ng, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, vol. 15 of *JMLR Workshop and Conference Proceedings*, pp. 215–223, JMLR W&CP, 2011.

[41] T. Dozat, "Incorporating Nesterov Momentum into Adam," tech. rep., Stanford University, 2015.

[42] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *CoRR*, vol. abs/1408.5093, 2014.

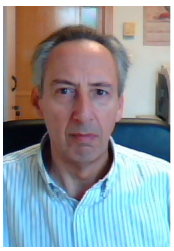
[43] F. Chollet, "Keras." https://github.com/fchollet/keras, 2015.

[44] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016.

Santiago López-Tapia received the degree in Computer Science and the Master degree Computer Science in 2014 and 2015, respectively. Currently, he is a Ph.D. student of the Visual Information Processing group, at the Department of Computer Science and Artificial Intelligence of the University of Granada.

Rafael Molina received the degree in mathematics (statistics) and the Ph.D. degree in optimal design in linear models from the University of Granada, Granada, Spain, in 1979 and 1983, respectively. He became Professor of Computer Science and Artificial Intelligence at the University of Granada, Granada, Spain, in 2000. He is the former Dean of the Computer Engineering School at the University of Granada (1992–2002) and Head of the Computer Science and Artificial Intelligence department of the University of Granada (2005–2007). His research interest focuses mainly on using Bayesian modeling and inference in problems like image restoration (applications to astronomy and medicine), superresolution of images and video, blind deconvolution, computational photography, source recovery in medicine, compressive sensing, low-rank matrix decomposition, active learning, fusion, and classification.

Prof. Molina serves as an Associate Editor of Applied Signal Processing (2005–2007); the IEEE Transactions on Image Processing (2010–2014); and Progress in Artificial Intelligence (2011–present); and an Area Editor of Digital Signal Processing (2011–present). He is the recipient of an IEEE International Conference on Image Processing Paper Award (2007) and an ISPA Best Paper Award (2009). He is a coauthor of a paper awarded the runner-up prize at Reception for early-stage researchers at the House of Commons.



Nicolás Pérez de la Blanca received the M.S. degree in mathematics from the University of Granada in 1975 and the Ph.D. degree from the same University in 1979. His research areas have involved, data analysis, biomedical image processing, pattern recognition and image classification. He has co-authored more than eighty research papers in these areas. Now is Full Professor at the Department of Computer Science and Artificial Intelligence of the University of Granada and is the head of the Visual Information Processing Group at this University. He has been president of the Spanish chapter of the IAPR (AERFAI) from 2000 to 2008. His current research interest areas are image and object classification, unsupervised learning and deep learning.